

# IT Security

## Multilevel Databases

Nikolaus Augsten

`nikolaus.augsten@sbg.ac.at`

Dept. of Computer Sciences  
University of Salzburg

Winter Semester 2015/16

# All infos about the database part in this lecture

<http://dbresearch.uni-salzburg.at/teaching/2015ws/its/>



# Table of Contents

- 1 MAC: Mandatory Access Control
- 2 MAC in Relational Databases
- 3 Oracle Label Security

# Table of Contents

- 1 MAC: Mandatory Access Control
- 2 MAC in Relational Databases
- 3 Oracle Label Security

# Mandatory Access Control (MAC)

- Why is discretionary access control (DAC) not enough?
  - users have the freedom to give other users access to data
  - all users see the same data (if they have access)
  - security policies cannot be centrally enforced
- Some applications need multilevel security
  - government, military, intelligence service
  - many industrial and corporate applications
- MAC is implemented in some DBMS (e.g., Oracle Label Security since 2009) or special versions of DBMS (e.g., SE-PostgreSQL)
- also operating systems implement MAC (SE-Linux, Windows Vista and later)

# MAC Basics

- Security classes: levels of trust
  - TS (top secret) > S (secret) > C (confidential) > U (unclassified, public)
- Subjects  $s$ 
  - users, roles, accounts, programs
  - clearance  $clear(s)$  is the trustworthiness of  $s$
  - $clear(s)$  is a security class
- Objects  $o$ :
  - data objects (e.g., relation, tuple, attribute values)
  - classification  $class(o)$  is the sensitivity of the data object
  - $class(o)$  is a security class

# Bell LaPadula

- Example of MAC used in database (and many other) systems
- Named after developers D. E. Bell and L. J. LaPadula
- Access control rules
  - **no read-up**:  $s$  is allowed to read  $o$  only if  $clear(s) \geq class(o)$
  - **no write-down**:  $s$  is allowed to write  $o$  only if  $clear(s) \leq class(o)$   
(also called \*-property)
  - **respect DAC**: respect discretionary access control rules
- **Trusted subjects**
  - must be trustworthy according to security policy
  - not restricted by the \*-property
  - can transfer data from higher to lower sensitivity

# Table of Contents

- 1 MAC: Mandatory Access Control
- 2 MAC in Relational Databases**
- 3 Oracle Label Security



# Multilevel Model

- Multilevel relation

- each attribute and each tuple in  $R(A_1, A_2, \dots, A_n)$  are classified
- $C_i = \text{class}(A_i)$  is an attribute classification
- $TC \geq \max\{C_i \mid 1 \leq i \leq n\}$  is the tuple classification
- the schema of the multilevel relation is

$$R(A_1, C_1, A_2, C_2, \dots, A_n, C_n, TC)$$

# Reading from Multilevel Relations

- Security requirement
  - users should not even know which data they cannot access
  - system should not reject requests for non-authorized data
  - but still the user should see a consistent view of the table
- Each clearance class  $c$  sees a different instance  $R^c$  of  $R$ :

$$R^c = (A_1^c, C_1^c, A_2^c, C_2^c, \dots, A_n^c, C_n^c, TC^c)$$

- Attributes  $A_i^c$  visible by  $s$  with  $clear(s) = c$ :
  - $A_i^c = A_i$  if  $C_i \leq c$
  - $A_i^c = \text{NULL}$  if  $C_i > c$
- Classifications  $C_i^c$  and  $TC^c$ :
  - $C_i^c = \min\{C_i, c\}$
  - $TC^c = \min\{TC, c\}$

# Reading from Multilevel Relations

- Security requirement
  - users should not even know which data they cannot access
  - system should not reject requests for non-authorized data
  - but still the user should see a consistent view of the table
- Each clearance class  $c$  sees a different instance  $R^c$  of  $R$ :

$$R^c = (A_1^c, C_1^c, A_2^c, C_2^c, \dots, A_n^c, C_n^c, TC^c)$$

- Attributes  $A_i^c$  visible by  $s$  with  $clear(s) = c$ :
  - $A_i^c = A_i$  if  $C_i \leq c$
  - $A_i^c = \text{NULL}$  if  $C_i > c$
- Classifications  $C_i^c$  and  $TC^c$ :
  - $C_i^c = \min\{C_i, c\}$
  - $TC^c = \min\{TC, c\}$

# How to Deal with Updates?

- Problem:
  - subject with low clearance sees NULL value and tries to change it
  - but this NULL value is due to the low clearance
- Option 1 (bad): update value
  - values of subjects with higher clearance get lost
  - writers do not even realize that they are doing something harmful
- Option 2 (bad): reject update
  - writing subject can infer that there is a sensitive non-NULL value
  - can be systematically exploited
- Option 3 (good): Polyinstantiation
  - maintain multiple versions of tuples
  - versioned tuples must differ by sensitivity class  $TC$
  - new model for integrity is required!

# Integrity in Multi-Level Databases

- Entity integrity
- Null integrity
- Inter-instance integrity
- Polyinstantiation integrity

# Entity Integrity

- Keys in instance  $R^c$  are called **apparent key**
- **Entity integrity**: for each  $R^c$  and for each tuple in  $R^c$ 
  1. key values must not be NULL
  2. all key attributes must have identical sensitivity class
  3. non-keys must be at least as sensitive as key

# Null Integrity

- Null integrity: for each  $R^c$ 
  1. NULL values always have sensitivity of key
  2. freedom of subsumption (= no unnecessary tuples)

# Inter Instance Integrity

- **Inter instance integrity:** for any pair  $R^c, R^{c'}$  with  $c' < c$

$$R^{c'} = f(R^c)$$

where  $f$  is called **filter**.

- The filter has the following properties
  1. for each tuple in  $R^c$  with key visible by  $c'$  a tuple must exist in  $R^{c'}$
  2. no other tuples exist in  $R^{c'}$
  3. subsumed tuples are eliminated



# Polyinstantiation Integrity

- Polyinstantiation integrity: uniqueness of tuples in  $R$ 
  - functional dependency

$$(key, C_{key}, C_i) \rightarrow A_i$$

must hold for any  $A_i$  in instance  $R^c$

# Implementation of Multilevel Databases

- integrity constraints allow implementation on top of “normal” relational system
- multilevel relation is fragmented into normal relations
- user queries compute answer from fragmented relations

# Table of Contents

- 1 MAC: Mandatory Access Control
- 2 MAC in Relational Databases
- 3 Oracle Label Security**

# Example: Oracle Label Security/1

- Label security provides MAC for Oracle DBMS
- Each tuple and each user has a label
- Labels consist of
  - level (class / clearance)
  - compartments: segregate data within a given level
  - groups: segregate data within level using organizational hierarchy
- subject  $s$  can access object  $o$  if
  - label of  $s$  must be at least level of  $o$
  - $s$  must have all compartments of  $o$
  - $s$  must have at least one group or supergroup of  $o$

# User labels

- User labels
  - max read clearance
  - min write clearance
  - default clearance (at login)
  - row level: default for inserted tuples
  - read and write compartments
  - read and write groups
- Trusted users / stored procedures
  - read / writeup / writedown
  - write across: change compartment and group
  - profile access: become other user (like Unix 'su')