Politiche mandatorie per l'integrità

Le politiche mandatorie per la segretezza controllano solo i flussi impropri di informazione $\Longrightarrow$ Non proteggono l'integrità.

Politica duale a quella della segretezza può essere applicata per l'integrità. $\Longrightarrow$ classificazione di soggetti e oggetti per l'integrità

Livello di integrità

- assegnato ad un utente riflette il livello di correttezza delle informazioni che l'utente ha e la fiducia nel fatto che l'utente non modificherà le informazioni impropriamente.
- assegnato ad un oggetto riflette il grado di fiducia nell'informazione contenuta nell'oggetto e il potenziale danno che potrebbe derivare dalla sua modifica o distruzione impropria.

Categorie definiscono aree di competenza di utenti e dati.

Modello di Biba per l'integrità

Definisce una politica mandatoria per l'integrità

Goal: prevenire il flusso delle informazioni verso classi più alte o non comparabili.

Politica di stretta integrità Basata su principi duali a quelli di BLP

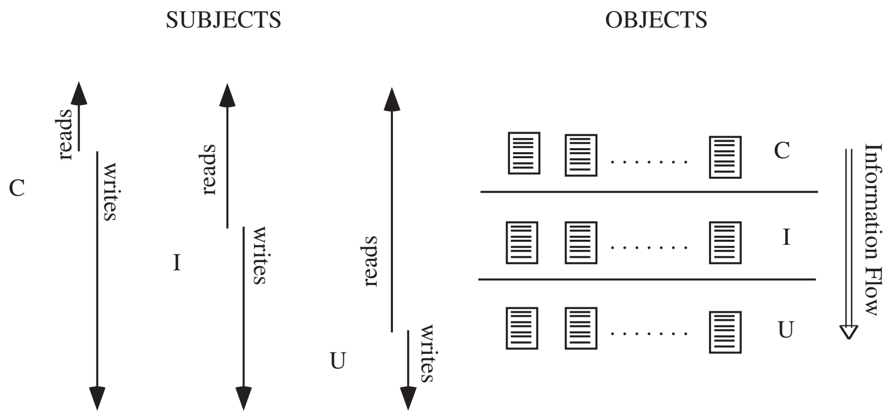
***-property** Un soggetto s può scrivere in un o solo se $\lambda(s) \succeq \lambda(o)$

simple property un soggetto s può leggere un oggetto o solo se $\lambda(o) \succeq \lambda(s)$

$\Longrightarrow$ NO WRITE UP
NO READ DOWN

Politiche di segretezza e integrità possono coesistere ...... **ma** ..... devono avere classificazioni "indipendenti"

Flusso di informazione per integrità



Modello di Biba – politiche alternative

Low-water mark per soggetti

- un soggetto s può scrivere un oggetti o solo se $\lambda(s) \succeq \lambda(o)$
- un soggetto s può leggere ogni oggetto o.
  Dopo l'accesso $\lambda'(s) := \mathsf{glb}(\lambda(\mathsf{s}), \lambda(\mathsf{o}))$.

Svantaggio: l'ordine delle operazioni modifica i privilegi del soggetto

Low-water mark per oggetti

- Un soggetto s può leggere un oggetto o solo se $\lambda(o) \succeq \lambda(s)$
- Un soggetto s può scrivere ogni oggetto o.
  Dopo l'accesso $\lambda(o) := \mathsf{glb}(\lambda(\mathsf{s}), \lambda(\mathsf{o}))$.

Svantaggio: non protegge l'integrità si limita a segnalare che è stata compromessa

Limitazione della politica di Biba

Controlla solo compromessi di integrità dovuti a flussi impropri.

L'integrità è un concetto più complesso.

Applicazione di politiche mandatorie a basi di dati

Il modello di Bell e La Padula è stato proposto per la protezione a livello di sistema operativo.

Approcci successivi hanno investigato la applicazione della politica mandatoria a modelli di gestione di dati (DBMS, sistemi ad oggetti, ...)

Mentre in sistemi operativi la classificazione è assegnata a livello di file, DBMS possono supportare un livello di granularità maggiore

- relazione
- attributo
- tupla
- elemento

Modello relazionale

Ogni relazione è caratterizzata da

- Schema della relazione $R(A_1, \ldots, A_n)$. Indipendente dallo stato.
- Istanza della relazione, dipendente dallo stato, composta da tuple $(a_1, \ldots, a_n)$

Name	**Dept**	**Salary**
Bob	Dept1	100K
Ann	Dept2	200K
Sam	Dept1	150K

Attributi chiave identificano univocamente le tuple

- Non ci possono essere due tuple con chiavi uguali
- Gli attributi della chiave non possono assumere valori nulli

Modello relazionale multilivello

In DBMS che supportano classificazione a livello di elemento, ogni relazione è caratterizzata da

- Schema della relazione $R(A_1, C_1, \ldots, A_n, C_n)$, indipendente dallo stato
  - $C_i, i = 1, \ldots, n$ intervallo di classi di accesso
- Insieme di istanze della relazione R_c dipendenti dallo stato; una istanza per ogni classe di access c. Ogni istanza è composta da tuple $(a_1, c_1 \ldots, a_n, c_n)$.

  L'istanza a livello c contiene solo elementi la cui classificazione è dominata da c.

Modello relazionale multilivello

Accesso controllato secondo i principi della politica mandatoria

- no read up (la vista di un soggetto ad una certa classe contiene solo gli elementi con classificazione dominata da quella classe)
- no write down però ulteriormente ristretto

  $\Longrightarrow$ Ognuno scrive al proprio livello

  Con granularità fine il write up non è necessario

Relazione multilivello – esempio

Name	λ_{N}	**Dept**	λ_{D}	**Salary**	λ_{S}
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S

Instanza U

Name	λ_{N}	**Dept**	λ_{D}	**Salary**	λ_{S}
Bob	U	Dept1	U	100K	U
Sam	U	Dept1	U	-	U

L'instanza S coincide con tutta la relazione

Modello relazionale multilivello

Per ogni tupla di una relazione multilivello

- tutti gli attributi chiave devono avere la stessa classificazione
  $\forall t \in T, \forall A_i, A_j \in AK : \lambda(t[A_i]) = \lambda(t[A_j])$
- la classificazione degli attributi non chiave deve dominare la classificazione degli attributi chiave
  $\forall t \in T, \forall A_i \in AK, A_j \notin AK : \lambda(t[A_j]) \geq \lambda(t[A_i])$

Poliinstanziazione

Il problema principale nel supportare classificazione a livello di granularità fine è l'introduzione della poliinstanziazione

- presenza simultanea di più oggetti con lo stesso nome ma diversa classificazione
  $\Longrightarrow$ differenti tuple con la stessa chiave ma
  - classificazione diversa per la chiave (tuple poliinstanziate)
  - valori e classificazione diversa per uno o più attributi (elementi poliinstanziati)

Poliinstanziazione

Tuple poliinstanziate

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Ann	U	Dept1	U	100K	U

Elementi poliinstanziati

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Sam	U	Dept1	U	100K	U

Poliinstanziazione

- Invisibile Un soggetto a basso livello inserisce dati in un campo che contiene già dati ad un livello maggiore o non comparabile
- Visibile Un soggetto ad alto livello inserisce dati in un campo che contiene già dati ad un livello minore

Poliinstanziazione invisibile

Un utente a basso livello chiede di inserire una tupla.

Esiste già una tupla con la stessa chiave primaria ma con classificazione maggiore

- comunicare all'utente l'esistenza della tupla $\Longrightarrow$ Information leakage
- sostituire la tupla esistente con la nuova tupla $\Longrightarrow$ integrità viene compromessa
- inserire la nuova tupla $\Longrightarrow$ tupla poliinstanziata

Tuple poliinstanziate – esempio

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S

Richiesta di un soggetto U

INSERT INTO Employee VALUES Ann,Dept1,100K

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Ann	U	Dept1	U	100K	U

Elementi poliinstanziati – esempio

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S

Richiesta di un soggetto U

UPDATE Employee SET Salary="100K" WHERE Name="Sam"

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	150K	S
Sam	U	Dept1	U	100K	U

Poliinstanziazione visibile

Un utente ad alto livello chiede di inserire una tupla.

Esiste già una tupla con la stessa chiave primaria ma con classificazione minore

- comunicare all'utente l'esistenza della tupla e rifiutare l'inserimento $\Longrightarrow$ denial of service
- sostituire la tupla esistente con la nuova tupla $\Longrightarrow$ information leakage
- inserire la nuova tupla $\Longrightarrow$ tupla poliinstanziata

Tuple poliinstanziate – esempio

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	U	Dept1	U	100K	U
Sam	U	Dept1	U	150K	S

Richiesta di un soggetto S

INSERT INTO Employee VALUES Ann,Dept2,200K

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	U	Dept1	U	100K	U
Sam	U	Dept1	U	150K	S
Ann	S	Dept2	S	200K	S

Elementi poliinstanziati – esempio

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	100K	U

Richiesta di un soggetto S

UPDATE Employee SET Salary="150K" WHERE Name="Sam"

Name	λ_N	**Dept**	λ_D	**Salary**	λ_S
Bob	U	Dept1	U	100K	U
Ann	S	Dept2	S	200K	S
Sam	U	Dept1	U	100K	U
Sam	U	Dept1	U	150K	S

Poliinstanziazione

Proposte di semantica di tuple e elementi poliiinstanziati

- tuple poliinstanziate $\Longrightarrow$ diverse entità del mondo reale
- elementi poliinstanziati $\Longrightarrow$ la stessa entità del mondo reale

.... poliinstanziazione però sfugge dal controllo.....

... probabilmente la ragione principale per cui i DBMS multilivello non hanno avuto successo

Prevenzione della poliinstanziate

Prevenzione di entità poliinstanziate:

- rendere visibili tutte le chiavi
- partizionare il dominio della chiave primaria
- permettere l'inserimento solo a soggetti *fidati*

Prevenzione di elementi poliinstanziati

- utilizzo di valori "restricted" (invece di vedere "null" i soggetti a basso livello vedono la presenza di un valore)
- una sola classificazione per tutti gli attributi chiave

Cover story

M-DBMS commerciali (es., Trusted Oracle) supportano classificazione a livello di tupla

La poliinstanziazione è probabilmente considerata la ragione principale del perchè i DBMS multilivello non hanno avuto successo

però la poliinstanziazione può essere utile

- cover story: informazione non corretta che il DBMS fornisce a soggetti a basso livello per proteggere informazione a livello maggiore

Supporto di classificazione a livello fine porta anche altri problemi

- es., supporto di vincoli di integrità è più complesso
- canali di inferenza

Architettura

- Trusted subject: dati a livelli differenti sono memorizzati in un solo database.

  Il DBMS deve essere trusted per assicurare l'obbedienza alla politica mandatoria.

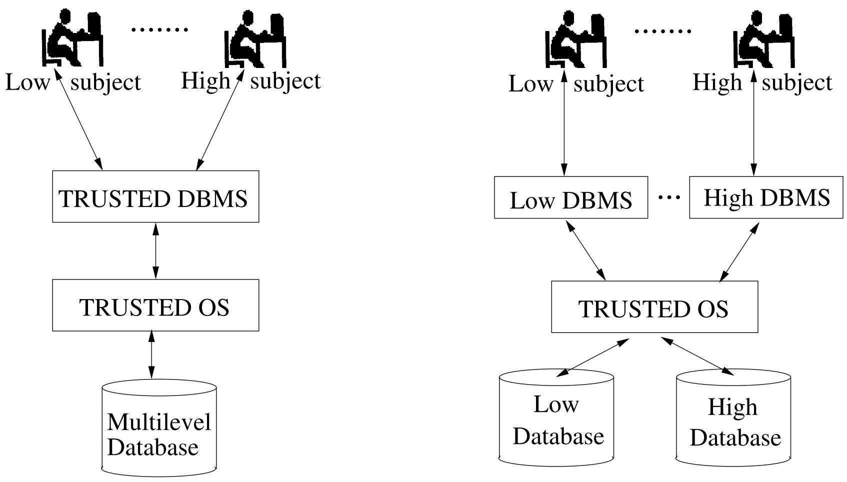
- Trusted computing base: dati sono partizionati in basi di dati differenti, uno per ogni livello.

  Solo il sistema operativo deve essere fidato.

  Ogni DBMS è confinato ai dati che i soggetti che usano quel DBMS possono accedere.

  (Necessità di algoritmi di decomposizione e ricostruzione).

Architettura



(a) Trusted subject

(b) Trusted computing base