

Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Інститут комп'ютерних наук та інформаційних технологій
Кафедра «Системи штучного інтелекту»



Лабораторна Робота №12
З предмету: «Організація баз даних та знань»

Виконав
студент групи КН-211
Турик Олександр
Прийняла :
Якимішин Х.М..

Львів-2020

Тема: “Розробка та застосування тригерів”

Мета роботи: Розробити SQL запити, які моделюють роботу тригерів: каскадне знищення, зміна та доповнення записів у зв’язаних таблицях.

Тригери бувають 6 видів: для 3 типів операцій (видалення, оновлення та вставлення даних), та для 2 станів (перед і після дією). У своїй роботі я використав 3 типи тригерів по одному на кожну операцію.

```
create trigger delete_company after  
delete on calendar.company for each row  
update calendar.participants set Position='Безробітний' where  
id_company=OLD.id_company;
```

Напишу запит для видалення компанії:

```
delete from calendar.company where id_company=6;
```

The screenshot displays a database management interface with two panels. The top panel shows a SQL query window with the following code:

```
46 • delete from calendar.company where id_company=6;  
47  
48 • select * from participants;  
49 • select * from company;
```

Below the query window is a table titled "company 54" with the following data:

id_company	Name	description
1	Softserve	SoftServe є однією з найбільших компаній-ро...
2	Epam	EPAM Systems — американська ІТ-компанія, ...
3	GlobalLogic	GlobalLogic Україна входить до GlobalLogic In...
6	N-IX	Українська ІТ компанія з головним офісом у Л...
NULL	NULL	NULL

The bottom panel shows the "Output" window with the "Action Output" tab selected. It displays the results of the SQL queries:

#	Time	Action	Message
1	19:25:01	select * from company LIMIT 0, 1000	4 row(s) returned

Below this, the same SQL queries are repeated:

```
46 • delete from calendar.company where id_company=6;  
47  
48 • select * from participants;  
49 • select * from company;
```

The bottom table shows the results of the second query, titled "participants 55":

id_Participant	Name	Surname	Position	E-mail	Password	id_company
1	Andy	Drevo	Design Lead	andronalin@gmail.com	123756	3
2	Oleg	Zhereb	Front End Deweloper	oleg@gmail.com	1234567	NULL
3	Masha	Rig	Python Deweloper	iammasha@gmail.com	qwerty	1
4	Petro	Petrenko	Senior Java	petrenko@gmail.com	ppetro12	3
8	Petro	Dmytrov	Front End Deweloper	yci-cipy+study@gmail.com	q1w2e3r4	6
NULL	NULL	NULL	NULL	NULL	NULL	NULL

Видалимо компанію і переглянемо нову інформацію про учасника:

```

46 • delete from calendar.company where id_company=6;
47
48 • select * from participants;
49 • select * from company;
50

```

	id_Participant	Name	Surname	Position	E-mail	Password	id_company
	2	Oleg	Zhereb	Front End Deweloper	oleg@gmail.com	1234567	NULL
	3	Masha	Rig	Python Deweloper	iammasha@gmail.com	qwerty	1
	4	Petro	Petrenko	Senior Java	petrenko@gmail.com	ppetro12	3
	8	Petro	Dmytrov	Безробітний	ydi-cipy+study@gmail.com	q1w2e3r4	6
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL

participants 56 x

Output

Action Output

#	Time	Action	Message
✓ 1	19:25:01	select * from company LIMIT 0, 1000	4 row(s) returned
✓ 2	19:25:39	select * from participants LIMIT 0, 1000	5 row(s) returned
✓ 3	19:26:26	delete from calendar.company where id_company=6	1 row(s) affected
✓ 4	19:26:34	select * from participants LIMIT 0, 1000	5 row(s) returned

Як бачимо, змінилася інформація про учасника у полі Position встановилося значення «Безробітний».

Наступний тригер я використав при додаванні нового користувача виконується шифрування пароль.

create trigger hide_password before insert
on calendar.user **for each row**

set new.password = **aes_encrypt**(new.password,'key-key');

Додам запит додавання нового користувача у таблицю user:

INSERT INTO `user`(`Name`, `Surname`, `E-mail`, `Password`, `Is_Admin`)
VALUES ('Юрій','Кривенчук','ykryvenchuk@gmail.com','*id*vech*',True);

```

69 • INSERT INTO `user`(`Name`, `Surname`, `E-mail`, `Password`, `Is_Admin`)
70 • VALUES ('Юрій','Кривенчук','ykryvenchuk@gmail.com','*id*vech*',True);
71
72 • desc event;
73 • select * from user;
74 • select * from event;

```

	id_User	Name	Surname	E-mail	Password	Is_Admin
	1	Наталя	Мельникова	nmelnykova@gmail.com	BLOB	0
	2	Наталя	Шаховська	nshakhovska@gmail.com	BLOB	1
	3	Роман	Гасько	rgasko@gmail.com	BLOB	0
	4	Христина	Якимшин	hyakymyshyn@gmail.com	BLOB	0
*	NULL	NULL	NULL	NULL	NULL	NULL

user 57 x

Output

Action Output

#	Time	Action	Message
✓ 1	19:33:41	create trigger hide_password before insert on calendar.user for each row set new.passw...	0 row(s) affected
✓ 2	19:34:07	select * from user LIMIT 0, 1000	4 row(s) returned

Створюю нового користувача і перегляну дані про нього:

The screenshot shows a SQL IDE interface with a query editor at the top and a result grid below it. The query editor contains the following SQL code:

```

69 • INSERT INTO `user`(`Name`, `Surname`, `E-mail`, `Password`, `Is_Admin`)
70 • VALUES ('Юрій', 'Кривенчук', 'ykryvenchuk@gmail.com', '*id*vech*', True);
71
72 • desc event;
73 • select * from user;
74 • select * from event;

```

The result grid displays the output of the queries. The first query (INSERT) is highlighted, showing the following data:

#	id_User	Name	Surname	E-mail	Password	Is_Admin
1	1	Наталія	Мельникова	nmelnykova@gmail.com	BL0B	0
2	2	Наталія	Шаховська	nshakhovska@gmail.com	BL0B	1
3	3	Роман	Гасько	rgasko@gmail.com	BL0B	0
4	4	Христина	Якимішин	hyakymyshyn@gmail.com	BL0B	0
28	28	Юрій	Кривенчук	ykryvenchuk@gmail.com	BL0B	1
*	NULL	NULL	NULL	NULL	NULL	NULL

The output pane at the bottom shows the execution log with the following entries:

#	Time	Action	Message
1	19:33:41	create trigger hide_password before insert on calendar.user for each row set new.passw...	0 row(s) affected
2	19:34:07	select * from user LIMIT 0, 1000	4 row(s) returned
3	19:36:14	INSERT INTO `user`(`Name`, `Surname`, `E-mail`, `Password`, `Is_Admin`) VALUES('Юрій'...	1 row(s) affected
4	19:36:20	select * from user LIMIT 0, 1000	5 row(s) returned

Тригер щодо оновлення я використав для поля Is_Admin у таблиці користувача: якщо змінити значення цього поля на 0 то всі події, створені користувачем стають локальними.

delimiter &&

create trigger update_position

after update on user *for each row*

begin

if new.is_Admin=0

then update (user *inner join* event) *SET* event.Status="Локальна"

where NEW.id_User=event.id_user;

end if;

end;&&

delimiter ;

Виконаю update таким запитом:

update user

set user.Is_Admin=0

where Name="Наталія" and Surname="Мельникова";

```

82 • update user
83 set user.Is_Admin=0
84 where Name="Наталія" and Surname="Мельникова";

```

	id_User	Name	Surname	E-mail	Password	Is_Admin
▶	1	Наталія	Мельникова	nmelnykova@gmail.com	BLOB	1
	2	Наталія	Шаховська	nshakhovska@gmail.com	BLOB	1
	3	Роман	Гасько	rgasko@gmail.com	BLOB	0
	4	Христина	Якимішин	hyakymyshyn@gmail.com	BLOB	0
	28	Юрій	Кривенчук	ykryvenchuk@gmail.com	BLOB	1
*	NULL	NULL	NULL	NULL	NULL	NULL

user 63 x Apply Revert

Output

Action Output

#	Time	Action	Message
✓ 1	19:46:39	create trigger update_position after update on user for each row begin if new.is_Admin=0 ...	0 row(s) affected
✓ 2	19:46:48	select * from user LIMIT 0, 1000	5 row(s) returned

Спочатку is_admin=1 і статус події яка посилається на id 1 коритсувача=“Глобальна”

```

73 • select * from user;
74 • select * from event;
75

```

	id_Event	Name	description	date	time	Link	Status	id_type	id_user
	1	IT-Arena	Найбільша IT подія Західної України	2020-09-20	13:00:00	itarena.com	Глобальна	NULL	3
	2	Захист проєктів	Захист семестрових проєктів	2020-05-18	10:00:00	NULL	Глобальна	NULL	2
	3	Здача лаб з бд	Дедлайн!!!	2020-05-19	10:00:00	NULL	Локальна	NULL	4
▶	4	Лекція з бд	Тест №4 на сервер huawei	2020-05-20	12:00:00	algotester.com	Глобальна	NULL	1
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

event 64 x Apply Revert

Output

Action Output

#	Time	Action	Message
✓ 1	19:46:39	create trigger update_position after update on user for each row begin if new.is_Admin=0 ...	0 row(s) affected
✓ 2	19:46:48	select * from user LIMIT 0, 1000	5 row(s) returned
✓ 3	19:50:16	select * from event LIMIT 0, 1000	4 row(s) returned

Запусти оновлення даних і перегляну події. Як можна побачити, зразу після видалення статус події змінився на “Локальна”

```

82 • update user
83   set user.Is_Admin=0
84   where Name="Наталія" and Surname="Мельникова";

```

	id_Event	Name	description	date	time	Link	Status	id_type	id_user
1	IT-Arena	Найбільша ІТ подія Західної України		2020-09-20	13:00:00	itarena.com	Глобальна	NULL	3
2	Захист проєктів	Захист семестрових проєктів		2020-05-18	10:00:00	NULL	Глобальна	NULL	2
3	Здача лаб з бд	Дедлайн!!!		2020-05-19	10:00:00	NULL	Локальна	NULL	4
4	Лекція з бд	Тест №4 на сервер huawei		2020-05-20	12:00:00	algotester.com	Локальна	NULL	1
	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

event 65 × Apply Revert

Output

Action Output

#	Time	Action	Message
✓ 2	19:46:48	select * from user LIMIT 0, 1000	5 row(s) returned
✓ 3	19:50:16	select * from event LIMIT 0, 1000	4 row(s) returned
✓ 4	19:51:49	update user set user.Is_Admin=0 where Name="Наталія" and Surname="Мельникова"	1 row(s) affected Rows matched: 1 Changed: 1 Warnings: 0
✓ 5	19:51:57	select * from event LIMIT 0, 1000	4 row(s) returned

Висновок: на даній лабораторній роботі я розробив SQL запити, які моделюють роботу тригерів: каскадне знищення, зміна та доповнення записів у зв'язаних таблицях.