ZAMYKÁNÍ DAT

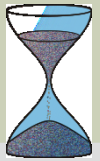
ZÁMKY

- Zabraňují tomu, aby více uživatelů měnilo v jednu chvíli stejná data
- Uživatel je získá automaticky a to vždy na nejnižší možné úrovni
- Nemění se (samy od sebe) v čase

Transakce 1

```
SQL> UPDATE employee
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

Transakce 2



```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3 WHERE employee_id=100;
```

ZPŮSOB ZAMYKÁNÍ

- Úrovně zamykání:
  - Řádkový zámek pro insert, update a delete
  - Žádný zámek pro select !!!
- Automatické řazení zámků do front
- Zámky zůstávají až do konce transakce (`COMMIT` nebo `ROLLBACK`)

Transakce 1

```
SQL> UPDATE employee
  2  SET salary=salary+100
  3  WHERE employee_id=100;
```

Transakce 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id=101;
```

KONKURETNÍ TRANSAKCE

Čas:	Transakce 1	`UPDATE hr.employees`
		`SET salary=salary+100`
		`WHERE employee_id=100;`

	Transakce 2	`UPDATE hr.employees`
		`SET salary=salary+100`
		`WHERE employee_id=101;`

09:00:00	Transakce 3	`UPDATE hr.employees`
		`SET salary=salary+100`
		`WHERE employee_id=102;`

	...	...

	Transakce *x*	`UPDATE hr.employees`
		`SET salary=salary+100`
		`WHERE employee_id=xxx;`

DML ZÁMKY

Transakce 1

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 107;
1 row updated.
```

Transakce 2

```
SQL> UPDATE employees
  2  SET salary=salary*1.1
  3  WHERE employee_id= 106;
1 row updated.
```

- Každá DML transakce potřebuje 2 zámky:
  - `EXCLUSIVE ROW LOCK` pro řádek, který měníme
    - Znemožňuje ostatním měnit stejná data
  - `TABLE (ROW) EXCLUSIVE LOCK` nad tabulkou, kterou měníme
    - Znemožňuje globální operace nad tabulkou, např. změnu struktury

ENQUEUE - FRONTY

- Enqueue mechanismus sleduje:
  - Uživatele čekající na nějaký zámek
  - Požadovaný typ zámku
  - Pořadí, ve kterém uživatelé na zámek čekají

KONFLIKTY ZÁMKŮ

Transakce 1	Čas	Transakce 2
`UPDATE employees SET salary=salary+100 WHERE employee_id=100;` `1 row updated.`	9:00:00	`UPDATE employees SET salary=salary+100 WHERE employee_id=101;` `1 row updated.`

`UPDATE employees SET COMMISION_PCT=2 WHERE employee_id=101;`
Čekáme na zámek, který drží transakce 2
	9:00:05
	…
	
		`SELECT sum(salary) FROM employees;`
		`SUM(SALARY)`
		`----------`
		`692634`

Stále čekáme!	16:30:00	Spousty dalších operací - select, insert, update a delete, ale žádný commit nebo rollback
`1 row updated.`	16:30:01	`commit;`

PŘÍČINY KONFLIKTU ZÁMKŮ

- Nepotvrzené změny
- Dlouhé tzv. long-running transakce
- Ruční vyvolání zámků na nepřiměřeně vysoké úrovni

DETEKCE KONFLIKTU ZÁMKŮ

- V dynamickém pohledu V$SESSION je položka BLOCKING_SESSION
- Podle ní pak můžeme najít ve stejném pohledu uživatele, který konflikt způsobil
- V dynamickém pohledu V$SQL pak najdeme i příslušný SQL příkaz

ŘEŠENÍ KONFLIKTU ZÁMKŮ

- Transakci, která zámek drží, můžeme:
  - Potvrdit (commit) –může jen uživatel, který ji provádí
  - Odvolat (rollback) –může jen uživatel, který ji provádí
  - Odpojit (kill) – může administrátor

- Syntaxe je následující:

```
ALTER SYSTEM KILL SESSION 'sid,serial#';
```

- sid i serial# najdeme v pohledu v$session

ŘEŠENÍ KONFLIKTU ZÁMKŮ

- **Ještě jednou celý postup:**

```
SQL> select SID, SERIAL#, USERNAME
from V$SESSION where SID in
(select BLOCKING_SESSION from V$SESSION)

       SID    SERIAL# USERNAME
---------- ---------- --------------------
       769        581 KIV

SQL> alter system kill session '769,581'
 immediate;
```

DEADLOCK

Transakce 1		Transakce 2

`UPDATE employees`
`SET salary = salary x 1.1`
`WHERE employee_id = 1000;`
	9:00	`UPDATE employees`
		`SET manager = 1342`
		`WHERE employee_id = 2000;`

`UPDATE employees`
`SET salary = salary x 1.1`
`WHERE employee_id = 2000;`
	9:15	`UPDATE employees`
		`SET manager = 1342`
		`WHERE employee_id = 1000;`

`ORA-00060:`
`Deadlock detected while`
`waiting for resource`
	9:16	

DOTAZY?