

Zámky, transakce a izolační úrovně v SQL Serveru

Mgr. David Gešvindr

MVP: Azure | MCSE: Data Platform | MCSD: Windows Store | MCT | MSP

david@wug.cz

 @gesvindr

Motivace

- Databázové systémy jsou častým úložištěm dat pro vaše aplikace
- Jejich správné použití je klíčové pro bezchybný chod aplikace
- Celá řada vývojářů pracuje s relační databází a není si plně vědoma:
 - Vlastností transakcí
 - Negativních důsledků souběžného zpracování
 - Možností, jak se problémům při souběžném zpracování bránit

Osnova

1. Transakce
2. Zámky
3. Izolační úrovně a problémy souběžného zpracování
4. Nešlo by to bez zámků?

Osnova

1. Transakce

2. Zámky

3. Izolační úrovně a problémy souběžného zpracování

4. Nešlo by to bez zámků?

Transakce

- Základní vlastnosti transakce:

Atomicity (atomičnost)

Consistency (konzistence)

Isolation (izolace)

Durability (trvalost)

Atomicity

- Všechny operace v rámci transakce musí být provedeny **kompletně** nebo **vůbec**
- Každá operace modifikující data probíhá automaticky v transakci
- Typy transakcí
 - Explicitní transakce
 - ◆ XACT_ABORT
 - Autocommit transakce
 - Implicitní transakce

Consistency

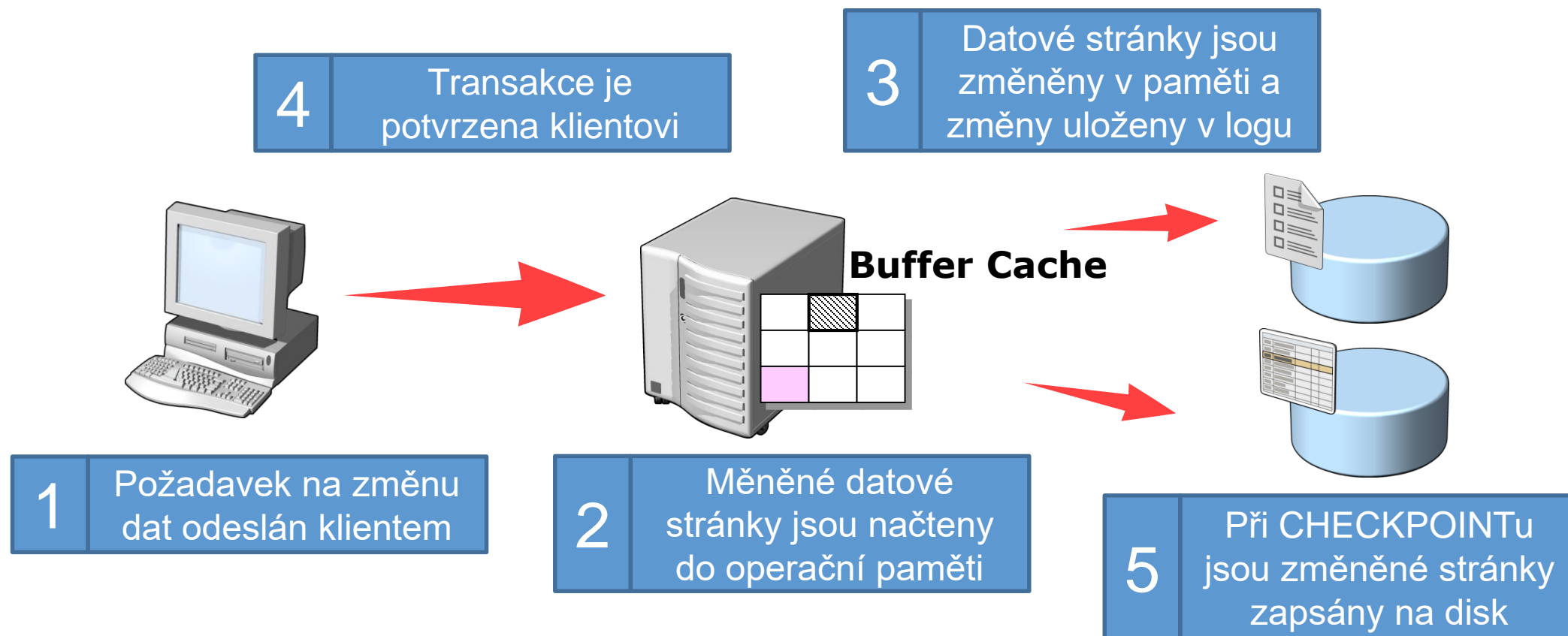
- Každá transakce může měnit data jen **povoleným způsobem**
- Databáze **musí být v platném** stavu před provedením transakce a i po provedení transakce
- Platný stav je určen integritními omezeními definovanými nad daty
 - Primary Key, Unique Key
 - Foreign Key
 - Check Constraint
 - Omezení operací nad daty – uložená procedura
- Také interní datové struktury musí být v platném stavu
 - Např.: Neobsahují ukazatele na neexistující stránky

Isolation

- Přesto, že probíhá v databázi více současných transakcí, **vzájemně se neovlivňují**
- Toto neplatí obecně
- Jsou různé úrovně izolace a výchozí úroveň izolace **read committed neřeší** všechny možné problémy
- Je nutné chápat rozdíly a správně nastavit izolační úroveň

Durability

- Jakmile klient obdrží informaci, že transakce byla potvrzena (committed), je **garantováno**, že změny jsou trvale uloženy



Osnova

1. Transakce

2. Zámky

3. Izolační úrovně a problémy souběžného zpracování

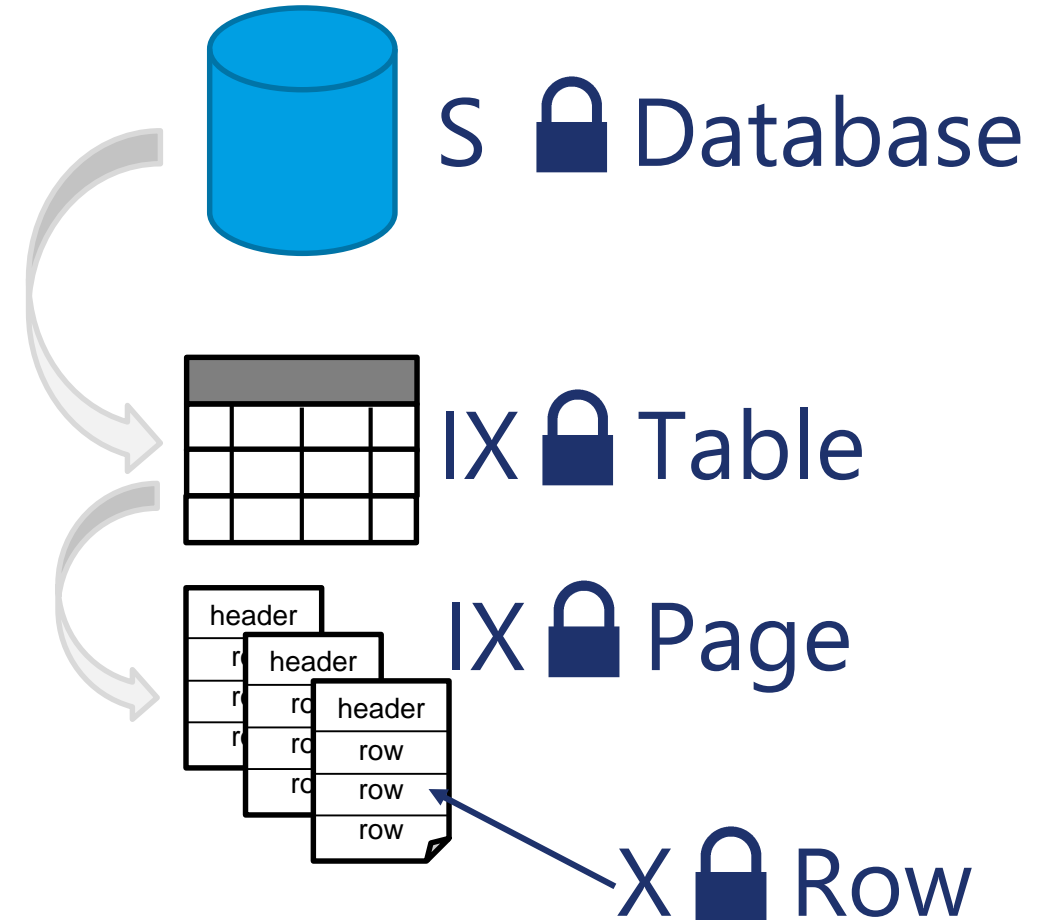
4. Nešlo by to bez zámků?

Pessimistic vs. optimistic concurrency

- Při souběžném přístupu může chtít více spojení **aktualizovat stejná data**
 - Aktualizace dat zahrnuje jejich přečtení a následnou změnu
- **Pessimistic concurrency**
 - Data jsou vhodně uzamykána, aby nemohla být změněna
- **Optimistic concurrency**
 - Data nejsou uzamykána, ale před každou aktualizací se kontroluje, že od načtení data nikdo nezměnil

Jak fungují zámky

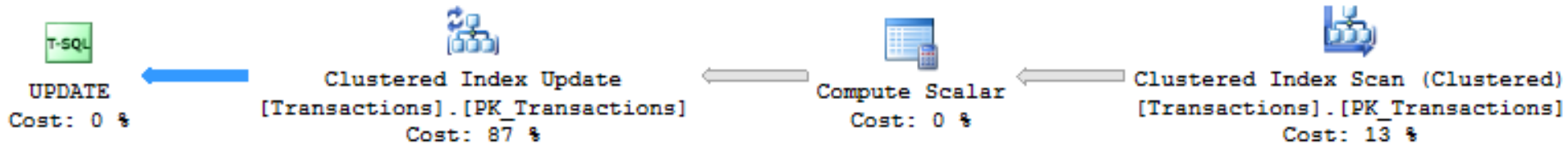
1. Připojíme se k databázi
 - Vznikne sdílený zámek nad databází
2. Spustíme dotaz:
`UPDATE table1`
`SET Name = 'David'`
`WHERE Id = 10`
 - Je třeba exkluzivně uzamknout řádek
 - Zamykat je však třeba shora
 - Intent eXclusive zámek



Typy zámků v SQL Serveru

- Shared Lock
 - Je možné zdroj číst, ale neměnit
- Update Lock
 - Signalizace, že budeme měnit, ale zatím neměníme
 - Kvůli snížení blokování čtení

		Aktivní zámek		
		Shared	Update	Exclusive
Požadovaný zámek	Shared	GRANT	GRANT	WAIT
	Update	GRANT	WAIT	WAIT
	Exclusive	WAIT	WAIT	WAIT

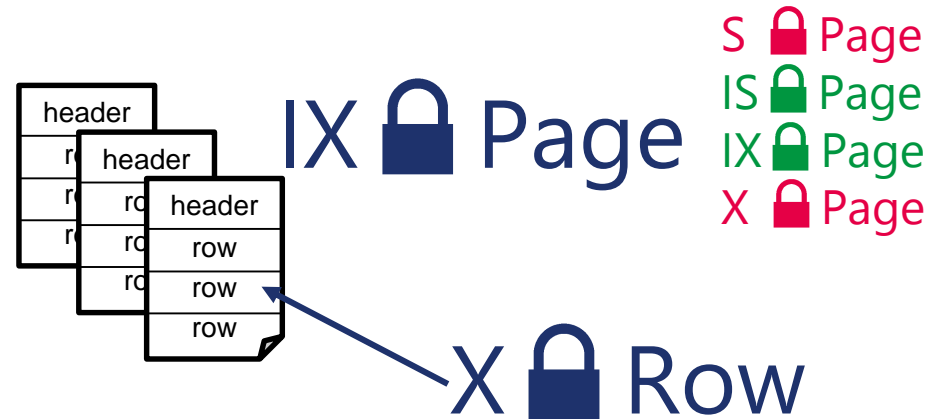
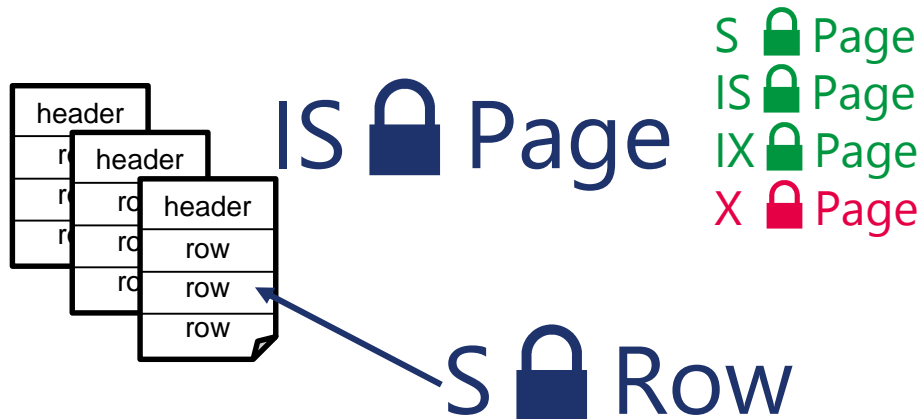


- Exclusive Lock
 - Je možné zdroj modifikovat

Typy zámků v SQL Serveru

- Intent Lock

- Chrání před vytvořením nekompatibilního zámku nad nadřazeným objektem



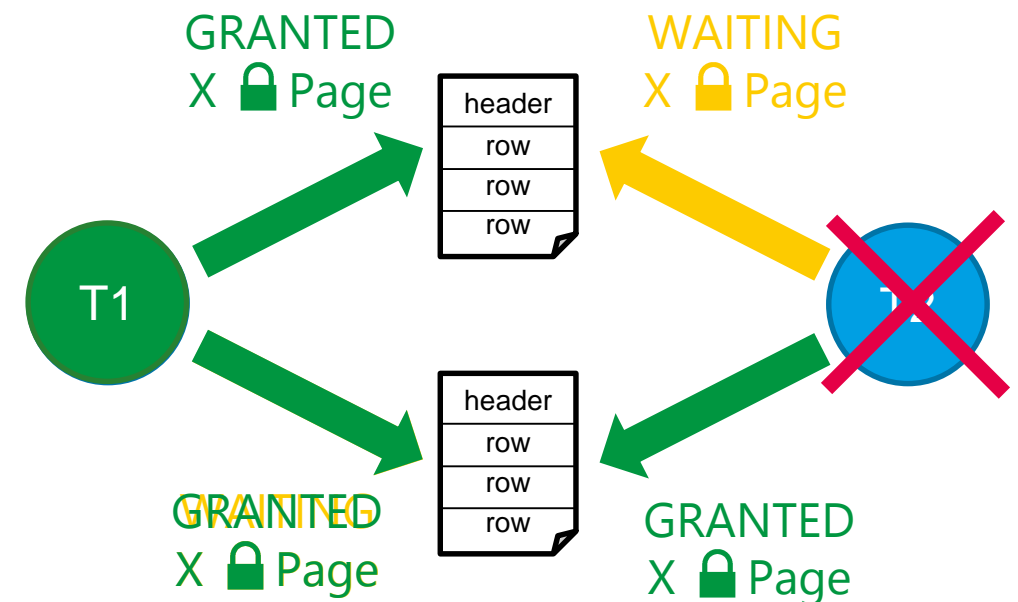
Rozsah zámků

- SQL Server určí při kompilaci dotazu potřebné rozsahy zámků
 - Snaha je co nejméně blokovat ostatní
- Lock Escalation
 - Pokud je při běhu dotazu překročena hranice 5000 zámků
 - Zvýšena úroveň zámků, sníženy nároky na zdroje



Live Lock a Deadlock

- Blocking (live locking)
 - Transakce čeká na uvolnění jiných zámeků
 - Transakce se nemohou předbíhat
- Deadlock
 - Uváznutí
 - Background deadlock monitor
 - Deadlock victim error 1205
 - Úprava aplikace
 - ◆ Přístup ke zdrojům ve stejném pořadí



Osnova

1. Transakce
2. Zámky
- 3. Izolační úrovně a problémy souběžného zpracování**
4. Nešlo by to bez zámků?

Lost Updates

- Vzájemné přepsání změn
- Např.:
 - **Editor A** si v 8:00 stáhne článek a celý den na něm pracuje
 - **Editor B** si v 9:00 stáhne článek a celý den na něm pracuje
 - **Editor A** uloží novou verzi článku v 15:00
 - **Editor B** přepíše článek svou verzí v 15:30
 - Práce **editora A** je ztracena
- Řešení:
 - **Editor A** si článek **exkluzivně uzamkne**

Dirty Read

- Čtení nepotvrzených změn (mezivýsledků) jiné transakce

```
/* Query 1 */  
SELECT age FROM users WHERE id = 1;  
/* will read 20 */
```

```
/* Query 2 */  
UPDATE users SET age = 21 WHERE id = 1;  
/* No commit here */
```

```
/* Query 1 */  
SELECT age FROM users WHERE id = 1;  
/* will read 21 */
```

```
ROLLBACK; /* lock-based DIRTY READ */
```

Level 0 – Read Uncommitted

Problém: Dirty Read

- Při čtení dat nejsou vytvářeny sdílené zámky
 - Transakce může číst i nepotvrzená data
- **Modifikace dat vytváří exkluzivní zámky vždy nehledě na izolační úroveň**
- Vhodné použití:
 - Čtení dat, kde nezáleží na přesném stavu
 - Není blokováno čtení a není blokován ani další zápis
 - ◆ Jsou pouze blokovány modifikace schématu (SCH_S)
 - **Používat velmi opatrně**

Level 1 – Read Committed

Inconsistent Analysis / Non-repeatable Reads + Phantoms

- Pro čtení dat jsou vyžadovány sdílené zámky
 - Není možné číst nepotvrzená data (exkluzivní zámky uvolněny na konci transakce)
- Neposkytuje absolutní přesnost při čtení dat
 - Při rozsáhlém čtení je i v rámci jedné tabulky možné narazit na nekonzistence
- Sdílené zámky jsou uvolňovány hned po přečtení dat

Non-repeatable Read

- Při opakovaném čtení dat v téže transakci přečteme jiná data

```
/* Query 1 */  
SELECT * FROM users WHERE id = 1;
```

```
/* Query 2 */  
UPDATE users SET age = 21 WHERE id = 1;  
COMMIT; /* in multiversion concurrency  
control, or lock-based READ COMMITTED */
```

```
/* Query 1 */  
SELECT * FROM users WHERE id = 1;  
COMMIT; /* lock-based REPEATABLE READ */
```

Level 2 – Repeatable Read

Phantoms

- Sdílené zámky jsou uvolněny až na konci transakce
- Ostatní transakce mohou data číst, ale nemohou je měnit
 - Jakákoliv přečtená data jsou po celou dobu života transakce stejná
- Nevýhody:
 - Jakákoliv data, kterých se dotkneme, uzamkneme až do konce transakce
 - Rizika rozsáhlého blokování
 - Problém s fantomy

Phantom Read

- Při opakovaném čtení dat se objeví nový záznam

```
/* Query 1 */  
SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;
```

```
/* Query 2 */  
INSERT INTO users VALUES ( 3, 'Bob', 27 );  
COMMIT;
```

```
/* Query 1 */  
SELECT * FROM users  
WHERE age BETWEEN 10 AND 30;  
COMMIT;
```


Level 3 – Serializable

- Využívá navíc **range-locks**
 - uzamknou určitý rozsah hodnot
 - nemůže být přidán žádný nový záznam
- Efektivní provedení **vyžaduje vhodný index**
 - Uzamčení vhodných uzlů ve stromu
 - **Pokud není vhodný index – vznikne zámek nad celou tabulkou**

Nastavení izolační úrovně

- Pro celé spojení pomocí příkazu:

```
SET TRANSACTION ISOLATION LEVEL READ UNCOMMITTED
```

- Izolační úroveň se od daného okamžiku aplikuje se na všechny operace prováděné **v daném spojení**

- Pro individuální operace s pomocí HINTů:

```
SELECT * FROM Users WITH(NOLOCK)
```

- Velmi dobře si rozmyslete jaký bude dopad, když budete míchat jednotlivé izolační úrovně uvnitř transakce
- Seznam HINTů: <https://msdn.microsoft.com/en-us/library/ms187373.aspx>

Osnova

1. Transakce
2. Zámky
3. Izolační úrovně a problémy souběžného zpracování
- 4. Nešlo by to bez zámků?**

Nešlo by to bez zámků?

- Od **SQL Serveru 2005** přidána možnost automaticky verzovat záznamy
- Rozšíření možností stávajících izolačních úrovní
- **Pozor – 2 oddělené technologie!**
 - **Read Committed Snapshot Isolation**
 - **Snapshot Isolation Level**

Read Committed Snapshot Isolation

- Zajištěna konzistence na úrovni **1 dotazu**
- Automaticky bude využíván místo vytváření sdílených zámků v READ COMMITTED izolační úrovni
- Aktivace pomocí příkazu:
`ALTER DATABASE [teched2015]
SET READ_COMMITTED_SNAPSHOT ON
WITH ROLLBACK AFTER 10`

Snapshot Isolation Level

- Nová izolační úroveň, kterou je třeba **explicitně aktivovat**
- Poskytuje vlastnosti úrovně **serializable**
- Minimalizuje počet zámků
 - Např.: Jeden uživatel generující report nezablokuje půl systému

- Aktivace pomocí příkazu:

```
ALTER DATABASE [teched2015]  
SET ALLOW_SNAPSHOT_ISOLATION ON
```

Negativní dopad snapshotů

- Aktivací jedné či obou úrovní snapshot izolace se automaticky začnou verzovat všechna modifikovaná data
- Zátěž na **tempdb** kde je uloženo **row version store**
- Hlavička řádku se zvětší o 14 bajtů, pokud se řádek verzuje
- Pokud 2 transakce ve **snapshot isolation level** začnou měnit navzájem data – dojde ke konfliktu a jedna je ukončena

Osnova

1. Transakce
2. Zámky
3. Izolační úrovně a problémy souběžného zpracování
4. Nešlo by to bez zámků?

Dotazy

Mgr. David Gešvindr

MVP: Azure | MCSE: Data Platform | MCSD: Windows Store | MCT | MSP

david@wug.cz

 @gesvindr