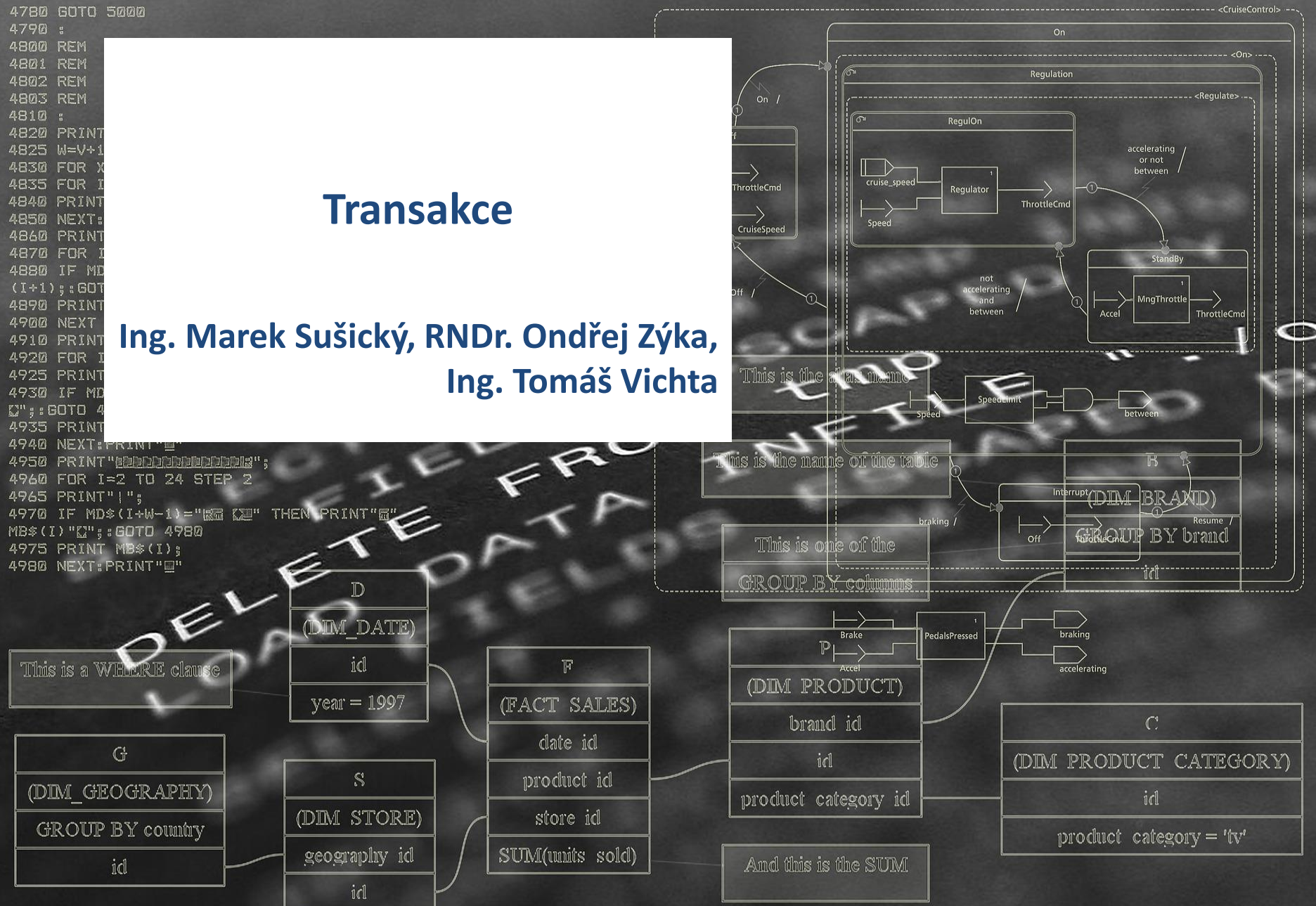
Transakce

Ing. Marek Sušický, RNDr. Ondřej Zýka,
Ing. Tomáš Vichta



Obsah

- Definice
- Savepoint, autonomní transakce
- Transakční módy
- Izolační úrovně
- Implementace pomocí zámků
- Implementace pomocí snapshotů
- Oracle, Microsoft SQL server
- Deadlock

Transakce

- Množina operací s daty, které splňují podmínku ACID
- Atomicity
- Consistency
- Isolation
- Durability

Transakce

- Atomicity
  - Vše nebo nic
  - I v případě chyby HW, software, aplikace, operačního systému.
  - Server potvrdí commit klientovi (response)

- Consistency
  - not null values
  - foreign key
  - unique constraint
  - check contraint
  - ...
  - Např. Oracle - deferrable constraints

Transakce

- Isolation
  - Ostatní nevidí necommitované změny
  - Já vidím i vlastní změny
  - Jako bych byl na serveru sám
- Durability
  - Trvalé uložení po commitu
  - Přežije jakoukoliv systémovou chybu

Transakce v jiných významech

- Transakce na aplikační úrovni
  - Vytvoření objednávky (desítky databázových transakcí),
  - Přepočet ohodnocení datového skladu (desítky minut).
- Transakce na podnikové úrovni
  - Schválení půjčky – několik aplikačních transakcí, workflow zasahující několik oddělení.
  - Možná interakce s uživateli
  - Není možný rollback – používá se opravný kód nebo papírový proces
  - Princip Eventual consistency

Savepoint

- ROLLBACK TO SAVEPOINT = vrácení změn dat
  - Není skok v kódu, ale "zapomenutí datových změn"
- **Nejdou** zrušit změny provedené před začátkem transakce

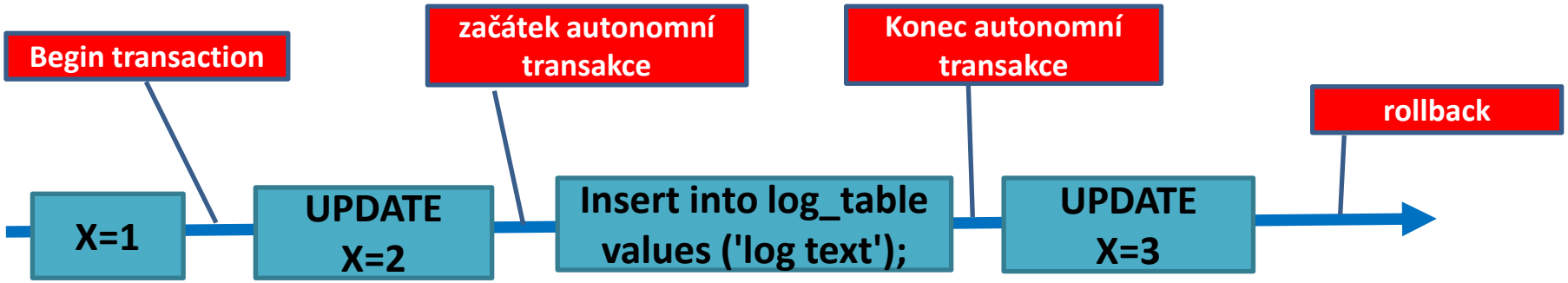
```
BEGIN TRANSACTION
  UPDATE TAB1 SET X=1, Z=10
SAVE TRANSACTION SAVEPOINT1
  UPDATE TAB1 SET X=2
ROLLBACK TRANSACTION SAVEPOINT1
  UPDATE TAB1 SET      Z=30
COMMIT
```

Po dokončení transakce

- X = 1
- Z = 30

Autonomní transakce

- Změna v datech mimo transakci
- Oracle specific
- V MS SQL např. pomocí linked serveru
- Používat obezřetně, nevhodné pro business logiku



- Po dokončení (rollback) transakce
  - X=1 - původní hodnota před zahájením transakce (rollback zafungoval)
  - ale, log_table obsahuje zápis s hodnotou 'log text' (nebyl rollbackován)

Konzistentní stav databáze

Dva přístupy

- Zamykání
  - Zamknu si data pro sebe
    - Tím blokuji ostatní!
  - Cizí transakce musí počkat na konzistentní stav DB
  - Zápis = nutný exkluzivní zámek
  - Čtení může exkluzivně zamykat - dle isolation level
- Multiversion concurency control (snapshots)
  - Snapshot konzistentního stavu k začátku transakce (nebo dotazu)
  - Commit se podaří první transakci => ztráta práce
  - Overhead s vytvářením snapshotů

- Oba přístupy
  - Hodně uživatelů a dlouhé transakce jsou problém

Transakční mód

- Tj. jakým způsobem/SQL příkazem zahájíme transakci
- Chained
  - Začátek transakce
    - První příkaz dávky
    - Insert, update, delete, select for update/holdlock, …
    - Nepoužívá se BEGIN TRANSACTION
  - Konec transakce
    - Commit, Rollback - vždy
- Unchained
  - Každý příkaz elementární transakce
  - Více příkazů v transakci
    - Nutný BEGIN TRANSACTION a COMMIT/ROLLBACK

- Kód musí být psaný pro konkrétní transakční mód
- Mód je možné měnit na úrovni připojení (pro některé servery)

Transakční mód

- Oracle
  - Chained mode
- MS SQL server, Sybase
  - Unchained i chained mód
- MySQL
  - SET autocommit = {0 | 1}

- Simulace (v klientských nástrojích)
  - Nástroj na pozadí generuje SQL příkazy pro ovládání transakcí
  - Simulace chained módu
    - Commit begin tran, rollback begin tran
    - Není ekvivalentní chained módu
  - Simulace unchained módu
    - Commit za každým příkazem

Porušení izolace transakcí

- Úplná (SERIALIZABLE) izolace
  - Náročná na výkon
  - Omezuje propustnost, paralelismus
- Typy porušení izolace
  - **Dirty write**
    - 2 transakce mění data současně = nulová izolace
  - **Dirty read**
    - Cizí transakce vidí necommitované změny
  - **Fuzzy read (Non-repeatable read)**
    - Stejný dotaz vrací stejný počet řádků, ale jiné hodnoty
    - Tj. vidím změny commitované po začátku mojí transakce
  - **Phantom**
    - Stejný dotaz vrací jiný počet řádků
    - Cizí transakce provedla INSERT, DELETE, nebo i UPDATE
    - Tj. též vidím změny commitované po začátku mojí transakce

ANSI isolation levels

- Izolační úrovně = jaká porušení izolace jsou přípustná
- Možnosti serveru
  - Čekat (zamykání)
  - Nelze-li splnit - server generuje chybu

	Dirty write	Dirty read	Fuzzy read	Phantom
Read uncommited	Not possible	Possible	Possible	Possible
Read commited	Not possible	Not possible	Possible	Possible
Repeatable read	Not possible	Not possible	Not possible	Possible
Serializable	Not possible	Not possible	Not possible	Not possible

Implementace pomocí zámků

- Typy zámků
  - Shared – transakce čte objekt, zámek se uvolní po načtení objektu nebo trvá do konce transakce
  - Exclusive – transakce mění objekt, zámek musí trvat do konce transakce
  - Update – objekt zřejmě bude změněn (cursor)
- Intent zámek
  - Příznak, že na části objektu je nastaven zámek
  - Intent table shared – shared zámek na tabulce informující, že je zamknutý nějaký řádek nebo stránka tabulky
- Rozsah zámků
  - Řádek, datová stránka, tabulka, schema, databáze
  - Data, Indexy, interní struktury
- Konkrétní implementace jsou složité

Kombinace zámků

- Microsoft® SQL Server 2012

jiná transakce může získat zámek ke stejnému zdroji

Existující zámky

	IS	S	U	IX	SIX	X	Sch-S	SCH-M	BU
Intent shared	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No
Shared	Yes	Yes	Yes	No	No	No	Yes	No	No
Update	Yes	Yes	No	No	No	No	Yes	No	No
Intent exclusive	Yes	No	No	Yes	No	No	Yes	No	No
Shared with intent exclusive	Yes	No	No	No	No	No	Yes	No	No
Exclusive	No	No	No	No	No	No	Yes	No	No
Schema stability	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes
Schema modify	No	No	No	No	No	No	No	No	No
Bulk update	No	No	No	No	No	No	Yes	No	Yes

Implementace pomocí zámků

- Často exkluzivně zamykaný řádek nelze číst
- Často čtený řádek nelze měnit
- Lock escalation
  - Zamknutí stránky, schématu, tabulky, ...
  - Zmenší režii zamykání
  - Ale zamkne více, než je potřeba
- V OLTP nutné krátké transakce
  - Deadlocky, Non serializable chyby, čekání
  - Exponenciální závislost na provozu

Speciální konstrukce

- Nečekat na uvolnění zámku nebo čekat maximálně určenou dobu

```
select * from author for update nowait;
select * from author for update wait 10;
```

- Číst jenom z neuzamčených oblastí

```
select * from author readpast;
```

- Manuálně uzamknout tabulku

```
lock table table_name
in {share | exclusive } mode
[ wait [ numsecs ] | nowait ]
```

Optimistické schéma zamykání (při Read Committed)

- Programová simulace serializable izolace
- Předpoklad nízké pravděpodobnosti konfliktu

Postup

- Načtu hodnoty (SELECT)
- Zpracovávám načtené hodnoty
- Ukládám a kontroluji, že nenastal konflikt
- Pokud ne, zařídím opravu
  - Opakovat v cyklu
  - Nebo chyba uživateli

- Nutná Statement level consistency
- Za vyšší výkon platíme prací programátora

Optimistické schéma zamykání - příklad

```
select @old_value=column01 from table01 where Id=999
set @new_value=ComputeSomething(@old_value)
update table01
  set column01 = @new_value
  where column01 = @old_value and Id=999
if @@ROWCOUNT = 1
  -- OK
else if @@ROWCOUNT = 0
  raiserror('...')
else
   ...
```

- Test, kolik se updatovalo řádků
  - Jeden řádek – hodnota ve sloupci column01 se nezměnila -> commit
  - Žádný řádek – hodnota se změnila -> oprava
  - Více řádků – jiná chyba

Pesimistické schéma zamykání

- Programová simulace serializable izolace
- Předpoklad vysoké pravděpodobnosti konfliktu

Postup

- Načtu a zamknu hodnoty
  - Možné dlouhé čekání na možnost uzamčení
  - Chybový stav při dlouhém čekání
- Zpracování hodnot
- Commit
- Ostatní čekají po dobu zpracování nebo obdrží chybovou hlášku

Pesimistické schéma zamykání - příklad

```
select @old_value=column01
  from table01
  where Id=999 with holdlock
(řádek je uzamčen)
```

Použití:

```
      @old_value
```

Natavení:

```
      @new_value
update table01
  set column01 = @new_value
  where Id=999
commit
```

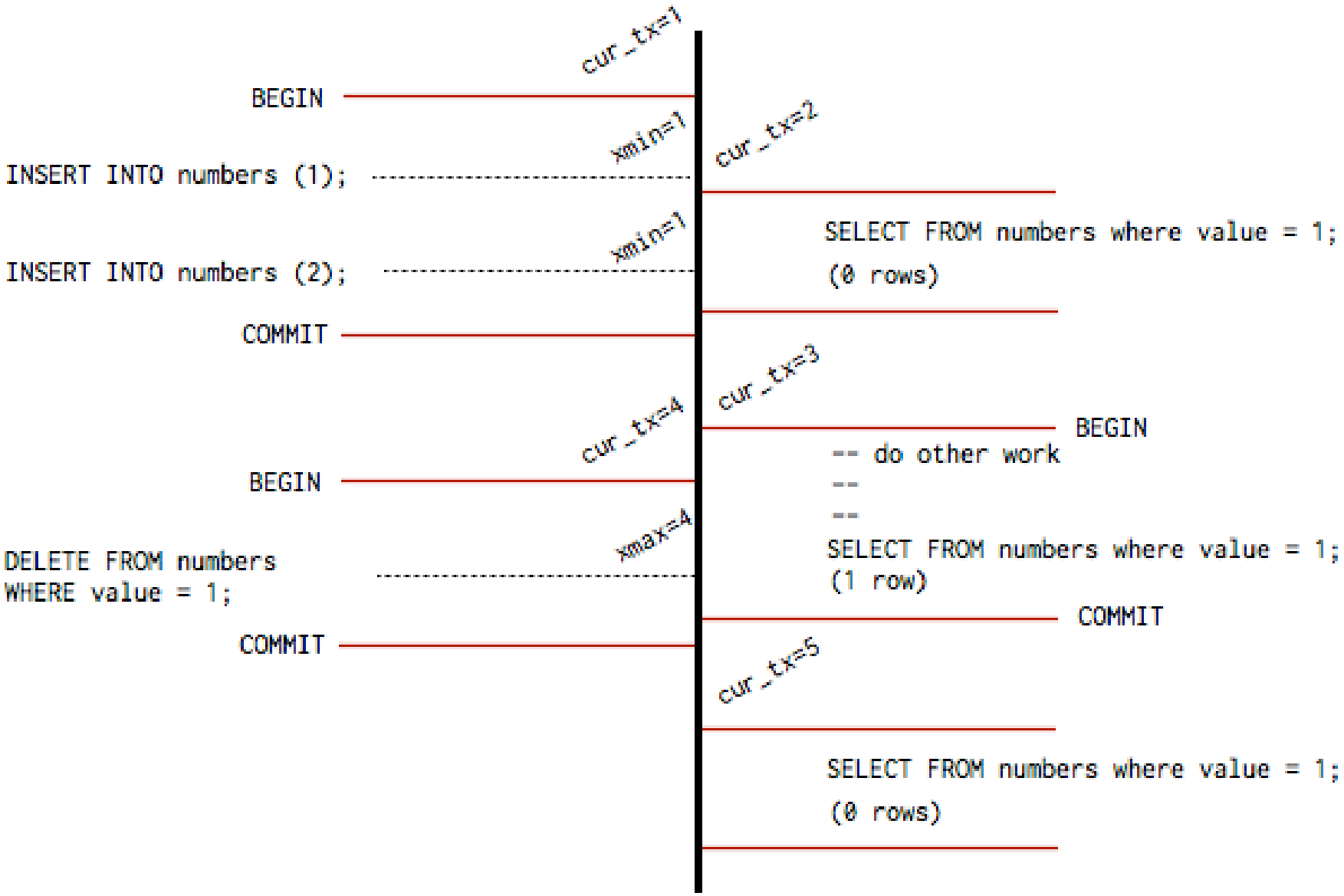
MVCC (PosgreSQL) - Multiversion concurrency control

- Každá transakce má rostoucí timestamp – XID (začátek transakce)
- Verze na úrovni řádků (obecně několik verzí)
  - xmin – timestamp vytvoření
    - timestamp (XID) transakce, která řádek vytvořila
  - xmax – timestamp ukončení
    - timestamp (XID) transakce, která řádek zrušila
  - update zruší starý a vytvoří nový řádek

- Transakce s timestampem x vidí řádky
  - Svoje změny
  - xmin < x, xmax neexistuje a transakce s xmin je komitovaná
  - xmin < x, xmax < x a transakce s xmax je nekomitovaná

- Podmínky pro změny (podle izolačních úrovní) – viz dokumentace.

MVCC


cur_tx=1
BEGIN
xmin=1
cur_tx=2
INSERT INTO numbers (1);
SELECT FROM numbers where value = 1;
xmin=1
INSERT INTO numbers (2);
(0 rows)
COMMIT
cur_tx=3
cur_tx=4
BEGIN
-- do other work
--
--
BEGIN
xmax=4
DELETE FROM numbers
WHERE value = 1;
SELECT FROM numbers where value = 1;
(1 row)
COMMIT
COMMIT
cur_tx=5
SELECT FROM numbers where value = 1;
(0 rows)

Izolační úrovně - Microsoft

```
SET TRANSACTION ISOLATION LEVEL
  { READ UNCOMMITTED
  | READ COMMITTED
  | REPEATABLE READ
  | SERIALIZABLE
  | SNAPSHOT
}
[ ; ]
ALTER DATABASE <database>
   SET READ_COMMITTED_SNAPSHOT ON
```

Izolační úrovně - Oracle

- Read Committed (Default)
  - konzistence na úrovni příkazu - statement level consistency
- Serializable Transactions
  - konzistence na úrovni transakce
- Read-only

```
SET TRANSACTION ISOLATION LEVEL READ COMMITTED;
SET TRANSACTION ISOLATION LEVEL SERIALIZABLE;
SET TRANSACTION ISOLATION LEVEL READ ONLY;
```

- Zámky pouze pro měněná data
- Zámky na úrovni řádků – neexistuje/není potřeba eskalace
- Transakce libovolně dlouhé – commit zatěžuje server

- Dlouhá Serializable transakce nevidí změny v datech, chyba v serializable módu se objeví až při příkazu porušení podmínek.

Deadlock v operačních systémech - Coffmanovy podmínky

- Procesy si navzájem blokují zdroje
- Může nastat při splnění všech Coffmanovo podmínek:
  - Mutual Exclusion – Prostředek může v jednom okamžiku vlastnit pouze jeden proces.
  - Hold and wait – Proces může žádat o další prostředky, i když má nějaké přiděleny.
  - No preemtion – Pokud proces prostředek vlastní, nelze mu ho bezpečně odejmout (musí ho vrátit sám).
  - Circual wait – Je možné uzavřít cyklus procesů vzájemně čekajících na zdroje svého předchůdce.
- V prostředí databází
  - Proces = transakce
  - Zdroj = zamknutelná entita (řádek, stránka, tabulka, ...)

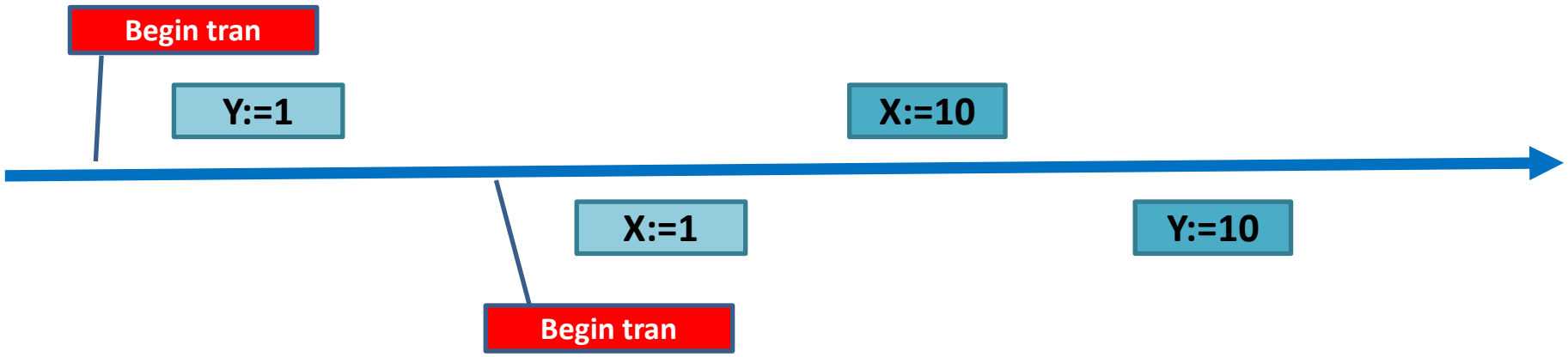
Deadlock – v databázích

- (Dvě) Transakce si navzájem blokují zdroje a čekají na jejich uvolnění.
- Vnější autorita (datový server)
  - Umí sledovat alokované zdroje
  - Vyřeší deadlock "poškozením" některého procesu
    - Zruší celý proces (transakci)
    - Přinutí proces, aby uvolnil nějaké zdroje
  - Servery řeší různě
    - výjimka
    - a/nebo rollback
    - a/nebo ukončení session

Deadlock

Server dokáže deadlock identifikovat

Server zruší jednu z transakcí



Deadlock

```
update publisher
   set name = 'Aldata
   Infosystems'
   where pub_id = '1389';
```

```
update publisher
   set name = 'New Age
   Books'
   where pub_id = '0736';
```

```
update publisher
   set name = 'New Age
   Books'
   where pub_id = '0736';
```

```
update publisher set name
   = 'Aldata Infosystems'
   where pub_id = '1389';
```

Chyba: ORA-00060

Předcházení deadlockům

- Nastavením všech zámků na začátku transakce
- Zamykání tabulek, stránek a řádků ve stejném pořadí
- Použití krátkých transakcí

V praxi většinou nelze plně vyřešit, nebo se to nevyplatí

- DB používá více (i cizích) aplikací
- Zamykat zdroje ve stejném pořadí reálně nelze
- S deadlockem je nutné počítat
  - ošetřit aplikačně
  - nebo alespoň nahlásit chybu
- Deadlocky nutno sledovat a optimalizovat

Non serializable (read committed)

```
set transaction isolation
   read committed;

var seatNum number

select Min(Number)
into seatNum
from Seat
where IsFree=1


update Seat
set IsFree=0
where SeatNum=seatNum


update reservation
set Seat=seatNum where ...

commit;
```

čas

```
set transaction isolation
read committed;



var seatNum number
select Min(Number)
into seatNum
from Seat
where IsFree=1


update Seat
set IsFree=0
where SeatNum=seatNum


update reservation
set Seat=seatNum where ...
commit;
```

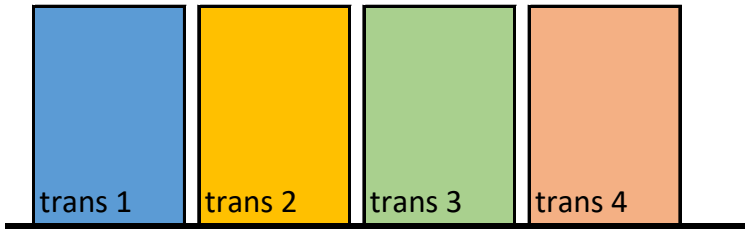
Serializable

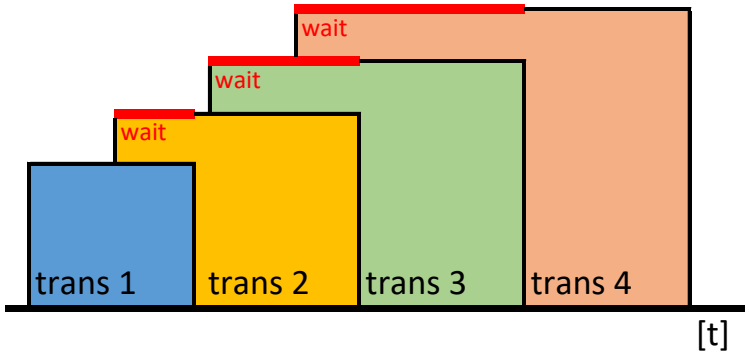
```
set transaction isolation
   read committed;

var seatNum number

select Min(Number)
into seatNum
from Seat
where IsFree=1


update Seat
set IsFree=0
where SeatNum=seatNum


update reservation
set Seat=seatNum where ...

commit;
```

čas

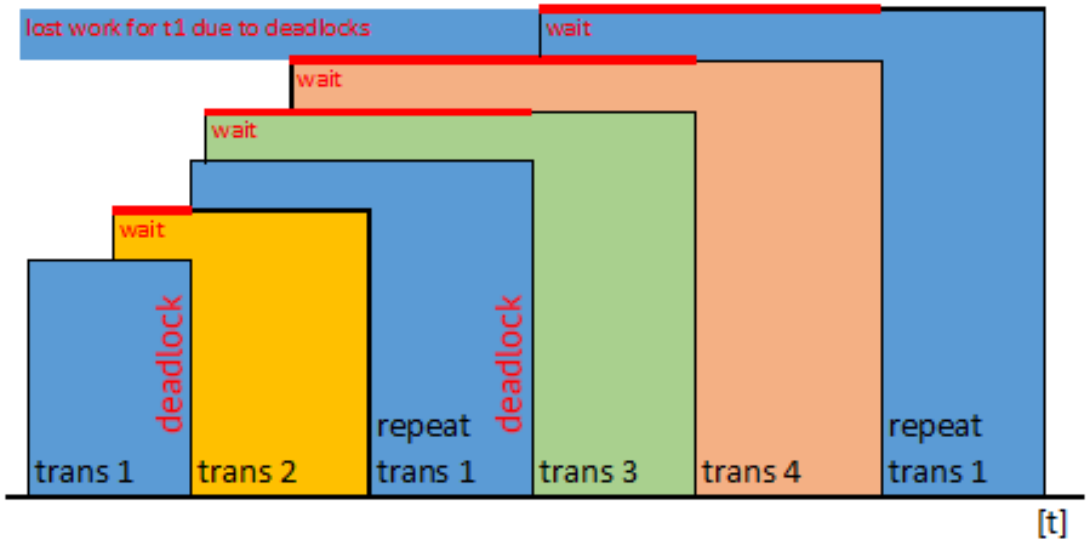
```
set transaction isolation
serializable;



var seatNum number
select Min(Number)
into seatNum
from Seat
where IsFree=1

update Seat
set IsFree=0
where SeatNum=seatNum

update reservation
set Seat=seatNum where ...

commit;
Chyba: ORA-08177
```

Contention





- Exponenciální chování, od kritického bodu systém de facto nefunkční
- Vliv čekání, popř. deadlocku a Non-serializable stavů, i opakovaně => ztracený čas a práce
- Na webu: uživatel zkouší nové requesty, ale původní stále běží a zatěžují servery

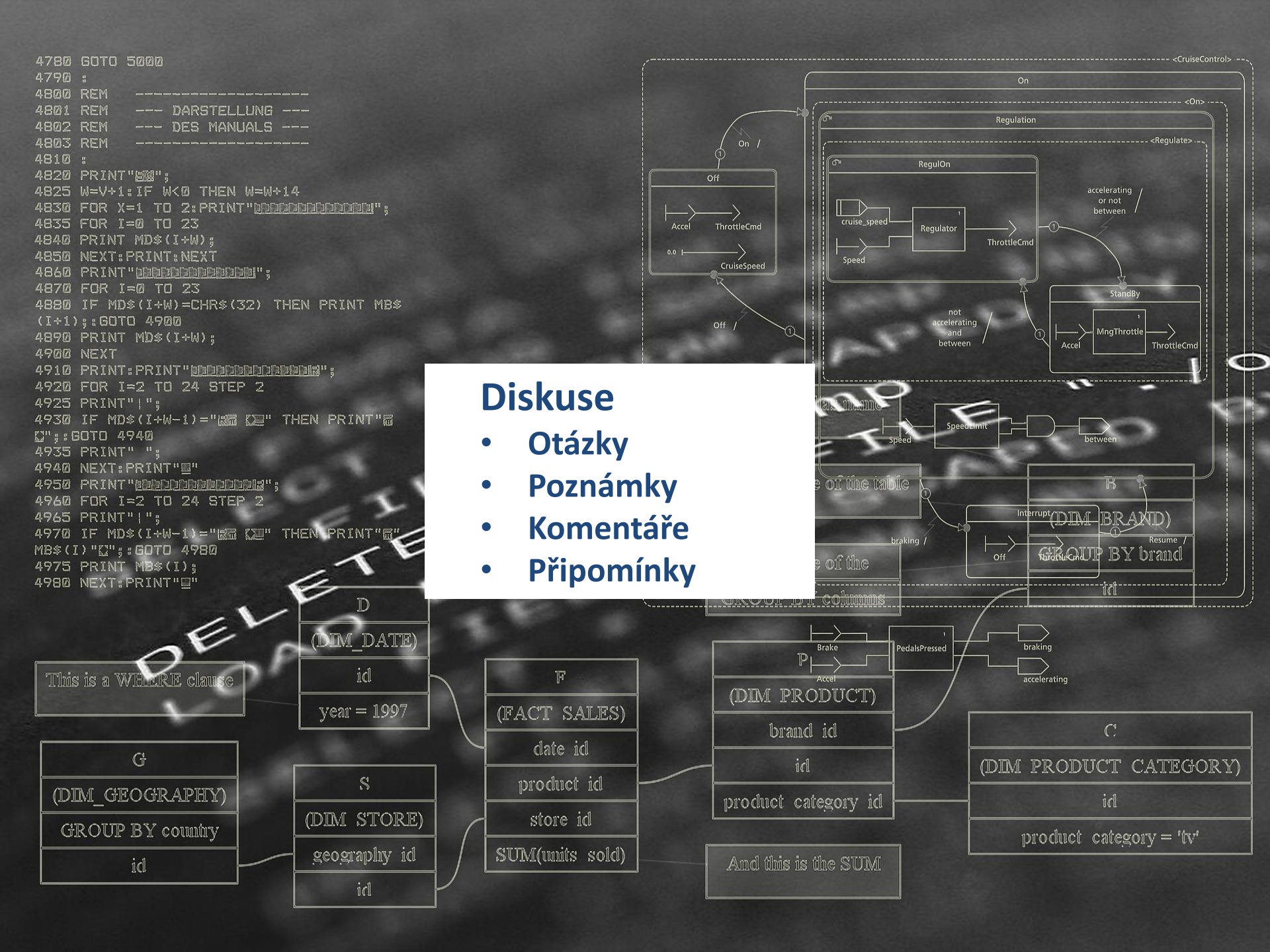


Závěry

- Složitost problematiky se projeví při zvýšení počtu uživatelů (paralelních transakcí), nikoliv při vývoji.
- Je třeba provádět cílené testy pro paralelní zpracování.
- Nárůst problémů je exponenciální s počtem paralelních transakcí.
- Commit může skončit s chybou a neprovést se.
- Je nutné znát přesné možnosti a chování konkrétního serveru (verze)

Co si zapamatovat

- Co to je transakce
- K čemu slouží savepoint
- Co to je autonomní transakce
- Jaké existují transakční módy a v čem se liší
- Definice izolační úrovně podle ANSI normy
- Jak se implementují isolační úrovně pomocí mechanismu zámků
- Jak se implementují isolační úrovně pomocí snapshotů
- Co to je optimistické a pesimistické schéma zamykání
- Co to je deadlock, jak se dá deadlockům předcházet



Diskuse

- Otázky
- Poznámky
- Komentáře
- Připomínky