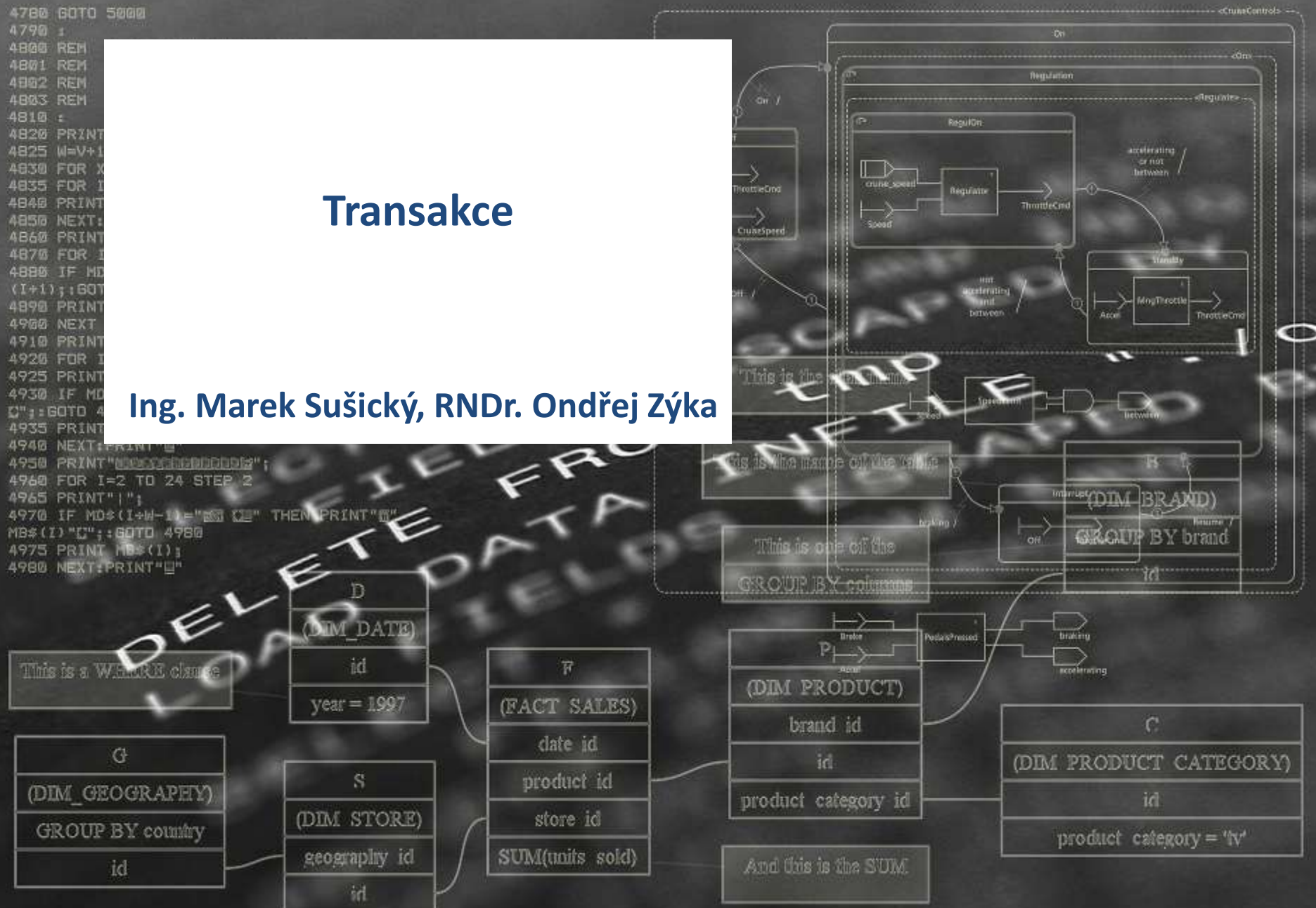
Transakce

Ing. Marek Sušický, RNDr. Ondřej Zýka



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
  - Změny provedené v transakci musí být zaneseny do databáze všechny (nebo žádná)
  - I v případě chyby hardware, chyby software, chyby aplikace, chyby operačního systému.
  - Uživatel musí být informován, zda se transakce uskutečnila a je ukončena.

- Consistency – po konci transakce musí být všechny požadavky na konzistenci databáze splněny
  - null values
  - foreign key
  - unique constraint

Transakce

- Isolation - Neukončené změny nejsou viditelné pro ostatní uživatele.
  - Uživatel provádějící změnu vidí i vlastní nekomitované změny.
  - Server každému uživateli dává pocit, že na serveru pracuje sám.

- Durability – Commitované transakce jsou trvale uložena v databázi.
  - Commitovaná znamená, že uživatel dostal informaci o ukončení commitu. (Příkaz commit byl ukončen a server předal řízení uživateli).
  - Takto uložené změny přežijí jakoukoliv systémovou chybu.

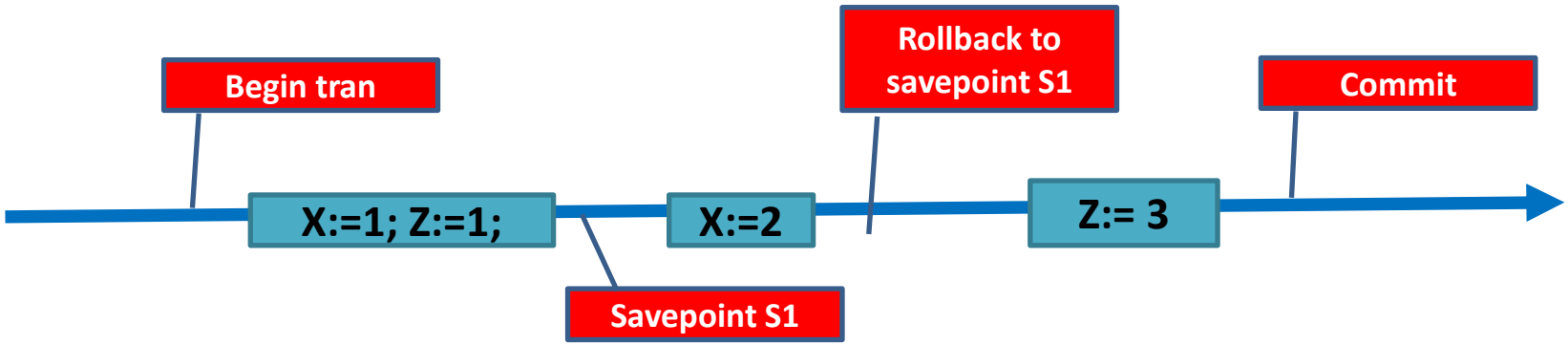
Transakce v jiných významech

- Transakce na aplikační úrovni
  - Vytvoření objednávky (desítky databázových transakcí),
  - Přepočet ohodnocení skladu (desítky minut).

- Transakce na podnikové úrovni
  - Schválení půjčky – několik aplikačních transakcí, workflow zasahující několik oddělení.
  - Možná interakce s uživateli
  - Není možný rollback – používají se opravný kód

Savepoint

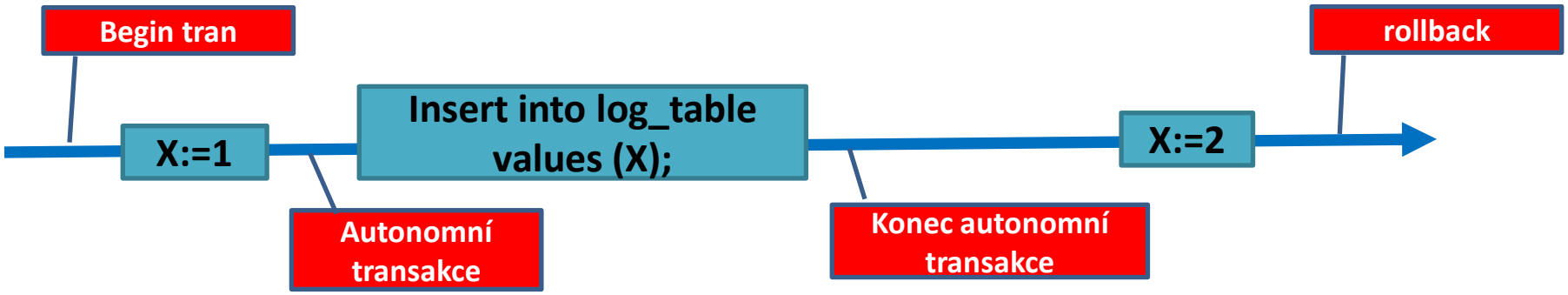
- Každé napojení do databáze – maximálně jedna transakce.
- Savepoint, rollback to savepoint
- V transakci se dají zrušit poslední provedené změny
- **Nejdou** zrušit pouze změny ze začátku transakce



- Po dokončení transakce
  - X = 1
  - Z = 3

Autonomní transakce

- Změna v datech mimo transakci
- Například zápis do logu
- Oracle
  - Na úrovní procedury, procedura obsahuje celou autonomní transakci



- Po dokončení (rollback) transakce
  - X původní hodnota
  - Zápis v tabulce s hodnotou 1

Konzistentní stav databáze

Dva přístupy

- Pokud A v transakci mění data, data jsou nekonzistentní. B musí počkat na konzistentní stav, na dokončení transakce A.
  - Implementace pomocí zamykání.
- I když A v transakci mění data, existuje poslední konzistentní stav před začátkem transakce. B může pracovat s tímto stavem.
  - Multiversion concurency control (snapshots)

- Pokud transakce A trvá dlouho
  - Zdržení transakce B
  - Práce s neaktuálními daty, chyby z neschopnosti udržet obrazy konzistentních stavů

Transakční mód

- Chained
  - Začátek transakce
    - První příkaz dávky
    - Insert, update, delete, select for update/holdlock, …
  - Konec transakce
    - Commit, rollback
- Unchained
  - Každý příkaz elementární transakce
  - Začátek transakce s více příkazy
    - Explicitně zadaný příkaz begin tran
  - Konec transakce
    - Commit, rollback

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

Transakční mód

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

- Simulace (v klientských nástrojích)
  - Chained módu
    - Commit begin tran, rollback begin tran
    - Není ekvivalentní chained módu
  - Unchained módu
    - Commit za každým příkazem

Porušení izolace transakcí

- **Zajištění úplné izolace transakcí je náročné na výkon a omezuje paralelní zpracování transakcí.**
- **Používají se následující definice porušení izolace transakcí.**

- **Dirty write**
  - Tranakce T1 změní data, která jsou měněna v probíhající transakci T2.
- **Dirty read**
  - Transakce T1 načte data změněná transakcí T2 ještě před tím, než transakce T2 provedla commit.
- **Fuzzy read (Non-repeatable read)**
  - Během transakce T1 se dvakrát načte řádek a pokaždé se vrátí jiná hodnota (transakce T2 modifikovala řádek mezi dvěma čteními).
- **Phantom**
  - Během transakce T1 se provede dvakrát stejný dotaz a pokaždé vrátí jiný výsledek (transakce T2 přidala nebo ubrala řádek použitý v dotazu transakce T1).

ANSI isolation levels

- Izolační úrovně jsou definováno tak, že zaručují zamezení výskytu porušení izolace transakcí.
- Zaručení může být vyřešeno dvojím způsobem:
  - Server čeká až bude moci zaručit požadované chování
  - Server generuje chybu, když zjistí, že není schopen zajistit požadované chování

	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 – transkce čte objekt, zámek se uvolní po načtení objeku nebo trvá do konce transakce
  - Exclusive – transakce mění objekt, zámek musí trvat do konce transakce
  - Update – objekt zřejmě bude změněn (cursor)
- Intend zámek
  - Příznak, že na části objektu je nastaven zámek
  - Intend table shared – existuje shared zámek na nějakém řádků nebo stránce
- Rozsah zámků
  - Řádek, datová stránka, tabulka, databáze
  - Data, Indexy, interní struktury
- Konkrétní implementace jsou složité

Kombinace zámků

- Microsoft® SQL Server 2012

Požadované zámky

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

- Problematické vlastnosti
  - Pokud transakce dlouho a často exklusivně zamykají řádek, nedá se řádek číst.
  - Pokud mnoho transakcí dlouhodobě zamyká řádek pro čtení, nedá se upravit.
  - Uzamknutí stránky nebo tabulky omezí přístup i na data, která transakce nepoužívá.
  - Režie s množstvím zámků
    - Zámky vyžadují zdroje datového serveru
    - Řešeno eskalací zámků na nadřazené datové struktury
  - Absolutní nutnost používat krátké transakce
    - Jinak uživatele čekají na dokončení cizích transakcí
    - Nebezpečí množství deadlocků

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

Způsob psaní aplikací v případě, že dochází zřídka ke konfliktům při přístupu k datům

- Načtení hodnot (a jejich zapamatování)
- Zpracování načtených hodnot
- Při zápisu výsledku se kontroluje, že původně načtené hodnoty jsou stále platné
- Pokud ne, provede se oprava
  - Znovu načti data a proveď opakuj výpočet
  - Informuj uživatele

Optimistické schéma zamykání - příklad

```
select @old_value=column01 from table01 where condition
(condition vybere jeden řádek)

Použití:
   @old_value
Natavení:
   @new_value

update table01
   set column01 = @new_value
   where column01 = @old_value
   and condition

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

Způsob psaní aplikací v případě, že ke konfliktům při přístupu k datům dochází často

- Načtení hodnot a jejich uzamčení
  - 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 condition
   with holdlock
(řádek je uzamčen)

Použití:
   @old_value
Natavení:
   @new_value

update table01
   set column01 = @new_value
   where condition

commit
```

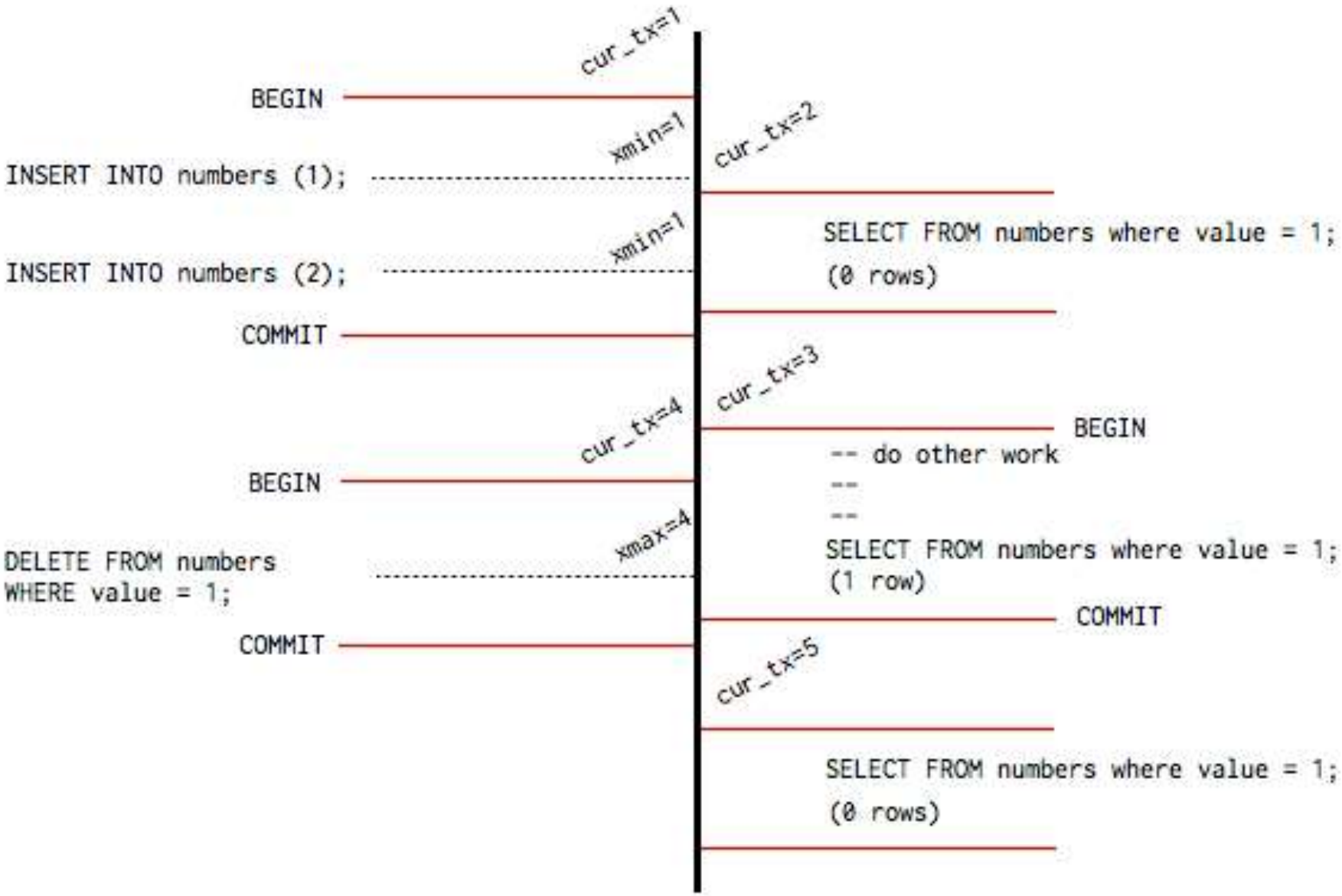
MVCC (PosgreSQL) - Multiversion concurrency control

- 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
  | SNAPSHOT
  | SERIALIZABLE
}
[ ; ]
```

Izolační úrovně - Oracle

- Read Committed (Default)
  - konzistence na úrovni příkazu
- 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 a čekají na jejich uvolnění.
- Situace může nastat v kterémkoliv systému, kde se více procesů dělí o prostředky.
- Přesněji: Může nastat v každém systému, kde mohou být splněny tzv. Coffmanovy podmínky:
  - 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.

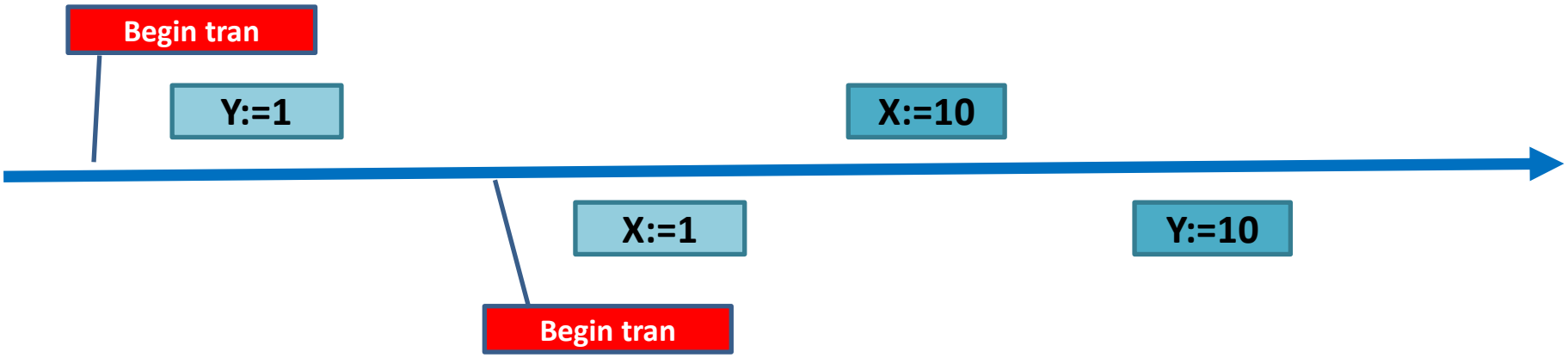
Deadlock – v databázích

- (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
  - Přinutí proces, aby uvolnil nějaké zdroje

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';
```

```
ORA-00060
```

Předcházení deadlockům

- Oslabením některé z Coffmanových podmínek
- Nastavením všech zámků na začátku transakce
- Zamykání tabulek ve stejném pořadí
- Použití krátkých transakcí

Non serializable

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

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

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

Serializable

```
set transaction isolation
   level serializable;

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

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

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

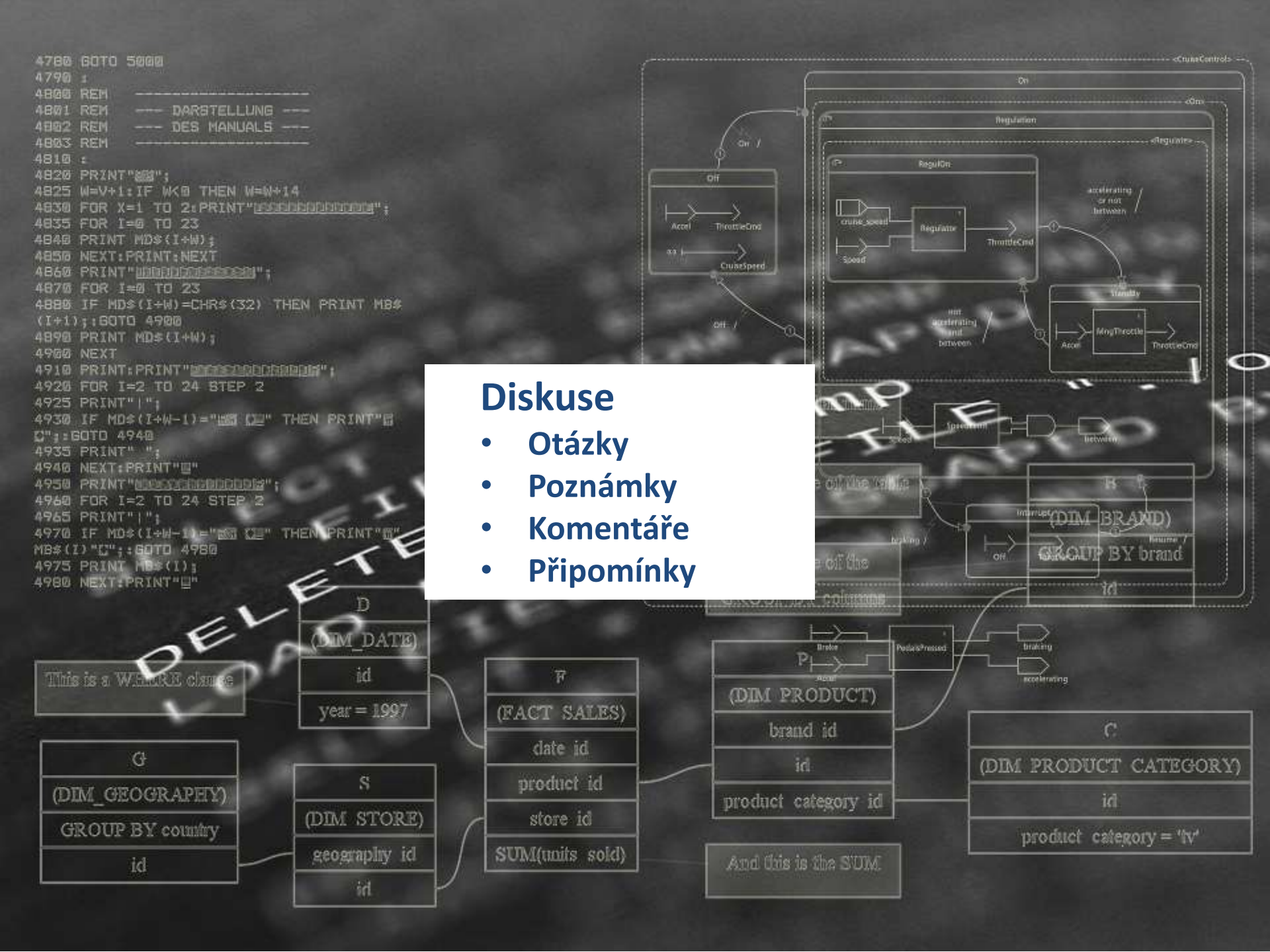
ORA-08177

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.
- Nutno počítat s tím, že transakce bude z nějakého důvodu zrušena při commitu.
- 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 isolační 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


4780 GOTO 5000
4790 :
4800 REM ----------
4801 REM --- DARSTELLUNG ---
4802 REM --- DES MANUALS ---
4803 REM ----------
4810 :
4820 PRINT"
4825 W=V+1:IF W<0 THEN W=W+14
4830 FOR X=1 TO 2:PRINT"
4835 FOR I=0 TO 23
4840 PRINT MD$(I+W);
4850 NEXT:PRINT:NEXT
4860 PRINT"
4870 FOR I=0 TO 23
4880 IF MD$(I+W)=CHR$(32) THEN PRINT MB$
(I+1);:GOTO 4900
4890 PRINT MD$(I+W);
4900 NEXT
4910 PRINT:PRINT"
4920 FOR I=2 TO 24 STEP 2
4925 PRINT"|";
4930 IF MD$(I+W-1)="
4935 PRINT" ";
4940 NEXT:PRINT"
4950 PRINT"
4960 FOR I=2 TO 24 STEP 2
4965 PRINT"|";
4970 IF MD$(I+W-1)="
MB$(I)"
4975 PRINT MB$(I);
4980 NEXT:PRINT"
Diskuse
Otázky
Poznámky
Komentáře
Připomínky
D
(DIM_DATE)
id
year = 1997
This is a WHERE clause
G
(DIM_GEOGRAPHY)
GROUP BY country
id
S
(DIM_STORE)
geography_id
id
F
(FACT_SALES)
date_id
product_id
store_id
SUM(units_sold)
P
(DIM_PRODUCT)
brand_id
id
product_category_id
And this is the SUM
C
(DIM_PRODUCT_CATEGORY)
id
product_category = 'tv'
(DIM_BRAND)
GROUP BY brand
id