



Тюрин Михаил

Уровни ИЗОЛЯЦИИ и репликация

Митап в Авито
сентябрь 2018



Зачем нам это?!

- ❖ важно, так как мерцающие баги
- ❖ репликация повсюду, так как в том числе Hot Standby с 9.0
- ❖ SSI — «true» serializable transactions с 9.1
- ❖ **bottom line:** еще раз окунуться в мир «аномалий» и БЫТЬ ГОТОВЫМИ дать ответ

локальный пример

<https://www.postgresql.org/docs/current/static/transaction-iso.html>

```
BEGIN;  
UPDATE website SET hits = hits + 1;  
-- run from another session: DELETE FROM website WHERE hits = 10;  
COMMIT;
```

assume website is a two-row table with website.hits equaling 9 and 10

<https://www.postgresql.org/docs/current/static/transaction-iso.html>

- ❖ две было: RC и RR (SI)
- ❖ добавили уровень «SSI» — *Serializable isolation*
- ❖ а как же мы живем то в RC (и в RR)?! — <https://www.postgresql.org/docs/current/static/explicit-locking.html#LOCKING-ROWS>

SSI

<https://wiki.postgresql.org/wiki/Serializable>

- ❖ ! не S2PL
- ❖ больше конкуренции
- ❖ но порядок «более хитрый»: The order in which transactions appear to have executed is determined by something **more subtle** than in S2PL

SSI

<https://wiki.postgresql.org/wiki/Serializable>

- ❖ https://wiki.postgresql.org/wiki/Serializable#R.26D_Issues
 - ❖ WAL file replay. While serializable implementations using S2PL can guarantee that the write-ahead log contains commits in a sequence consistent with some serial execution of serializable transactions, SSI cannot make that guarantee... **WAL replay will be at snapshot isolation** even for serializable transactions.
...
 - ❖ External replication. Look at how this impacts external replication solutions...

Начнем!

еще раз!

- ❖ <https://www.postgresql.org/docs/current/static/hot-standby.html#HOT-STANDBY-CAVEATS>
 - ❖ The Serializable transaction isolation level is not yet available in hot standby. An attempt to set a transaction to the serializable isolation level in hot standby mode will generate **an error**.
- ❖ <https://www.postgresql.org/docs/current/static/applelevel-consistency.html#SERIALIZABLE-CONSISTENCY>
 - ❖ красный **ВАРНИНГ**
 - ❖ This level of integrity protection using Serializable transactions does not yet extend to hot standby mode. Because of that, those using hot standby may want to use Repeatable Read and **explicit locking on the master**. // см. слайд 5

Отличный пример

❖ https://wiki.postgresql.org/wiki/SSI#Read_Only_Transactions

```
create table control
(
    deposit_no int not null
);
insert into control values (1);
create table receipt
(
    receipt_no serial primary key,
    deposit_no int not null,
    payee text not null,
    amount money not null
);
insert into receipt
    (deposit_no, payee, amount)
values ((select deposit_no from control), 'Crosby', '100');
insert into receipt
    (deposit_no, payee, amount)
values ((select deposit_no from control), 'Stills', '200');
insert into receipt
    (deposit_no, payee, amount)
values ((select deposit_no from control), 'Nash', '300');
```


https://wiki.postgresql.org/wiki/SSI#Read_Only_Transactions

session 1

```
begin; -- T1
insert into receipt (deposit_no, payee, amount) values
(
  (select deposit_no from control), 'Young', '100'
);
```

```
select * from receipt;
receipt_no | deposit_no | payee  | amount
-----+-----+-----+-----
          1 |          1 | Crosby | $100.00
          2 |          1 | Stills | $200.00
          3 |          1 | Nash   | $300.00
          4 |          1 | Young  | $100.00
```

```
commit;
ERROR:  could not serialize access
        due to read/write dependencies
        among transactions
DETAIL:  Cancelled on identification
        as a pivot, during commit attempt.
HINT:   The transaction might succeed if retried.
```

ТОЛЬКО
SSI

session 2

```
begin; -- T2
select deposit_no from control;
deposit_no
-----
          1

update control set deposit_no = 2;
commit;
```

```
begin; -- T3
select * from receipt where deposit_no = 1;
receipt_no | deposit_no | payee  | amount
-----+-----+-----+-----
          1 |          1 | Crosby | $100.00
          2 |          1 | Stills | $200.00
          3 |          1 | Nash   | $300.00
```

T1 -> T2 -> T3 -> T1

https://wiki.postgresql.org/wiki/SSI#Read_Only_Transactions

session 1

`rollback;`

`begin; -- T1 retry`

```
insert into receipt
(deposit_no, payee, amount)
values
(
(select deposit_no from control),
'Young', '100'
);
```

```
select * from receipt;
receipt_no | deposit_no | payee  | amount
-----+-----+-----+-----
          1 |           1 | Crosby | $100.00
          2 |           1 | Stills | $200.00
          3 |           1 | Nash   | $300.00
          5 |           2 | Young  | $100.00
```

`commit;`

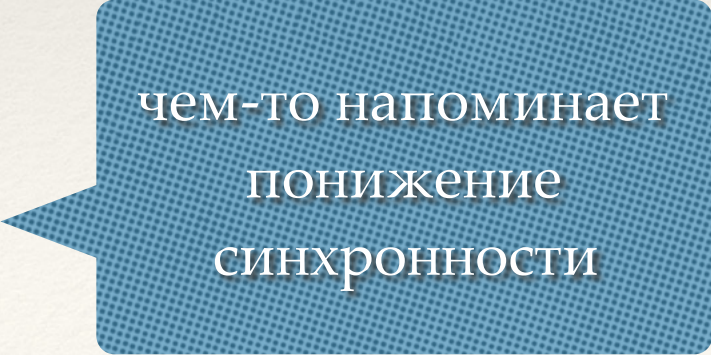
session 2

`commit;`

This would have been OK anytime after T3's SELECT.

https://wiki.postgresql.org/wiki/SSI#Read_Only_Transactions

- ❖ а на реплике мы не получим SSI, максимум SI — RR
- ❖ реплика будет применять коммиты в порядке их получения
- ❖ и чтение на ней **НЕ** войдет в «зависимости», так как на реплике нет информации о выполнении T1 в момент коммита T2
- ❖ реплика не получает такой информации
- ❖ итог: эффект **понижения** изоляции



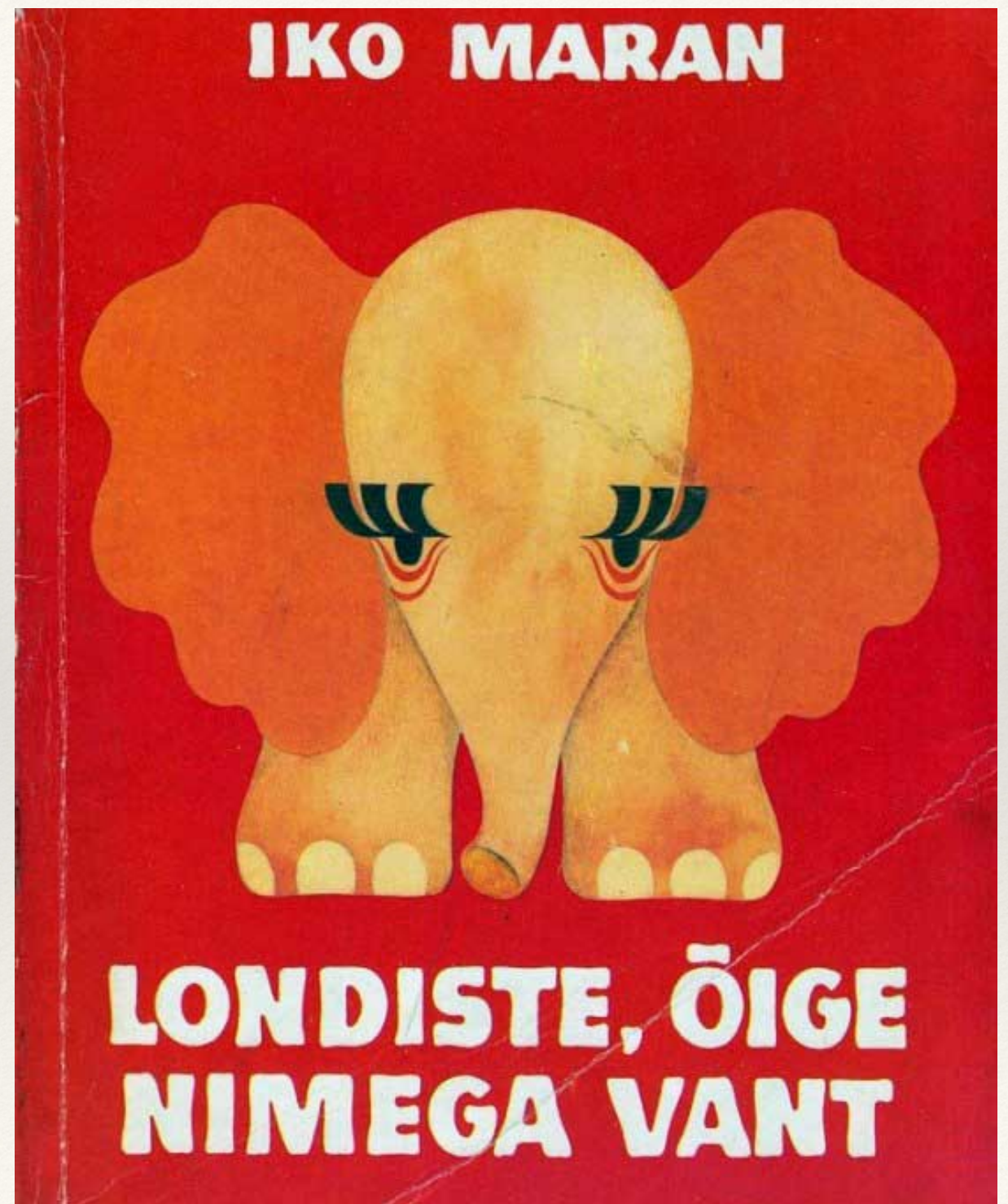
чем-то напоминает
понижение
синхронности

Логическая репликация

- ❖ встроенная с 9.7 — аналогично как с WAL
- ❖ триггерная: `londiste` и `slony`
 - ❖ ?
 - ❖ ?

Londiste

- ❖ аналогично — порядок прихода событий определяется порядком коммитов
- ❖ много есть докладов, в том числе от Авито



Londiste — pgq

```
SELECT * FROM queue q WHERE
```

```
/*вспомогательный скан-фильтр*/
```

```
q.ev_txid BETWEEN xmin1 AND xmax2    -- xmin1 = txid_snapshot_xmin(snap1), xmax2 = txid_snapshot_xmax(snap2)
```

```
/*вспомогательный скан-фильтр*/
```

```
/*суть*/
```

```
AND NOT is_visible( q.ev_txid, snap1 )  -- txid_visible_in_snapshot
```

```
AND      is_visible( q.ev_txid, snap2 )  -- txid_visible_in_snapshot
```

```
/*суть*/
```

```
----- и снэпшоты!
```

```
select pgq.ticker() -- txid_current_snapshot
```

Slony

- ❖ The name "slony" comes from the Russian word "слоны" which means «elephants». This is a reference to the PostgreSQL elephant logo, as well as being a "tip of the hat" to Vadim Mikheev, who came up with some of the core ideas Slony-I uses to work.
- ❖ slony — is the plural word for elephants, and indicates that a cluster consists of multiple databases
- ❖ slon — is the singular word for elephant; each replication node is managed by a program named "slon"
- ❖ slonik — is the word for a "little elephant," and is the name of the program used to configure the cluster. In effect, the "little elephant" tells the cluster, "here's what you need to do!"
- ❖ аналогічно ?
- ❖ — да! // Миша сказав

Slony

❖ <http://slony.info/images/Slony-I-implementation.pdf>

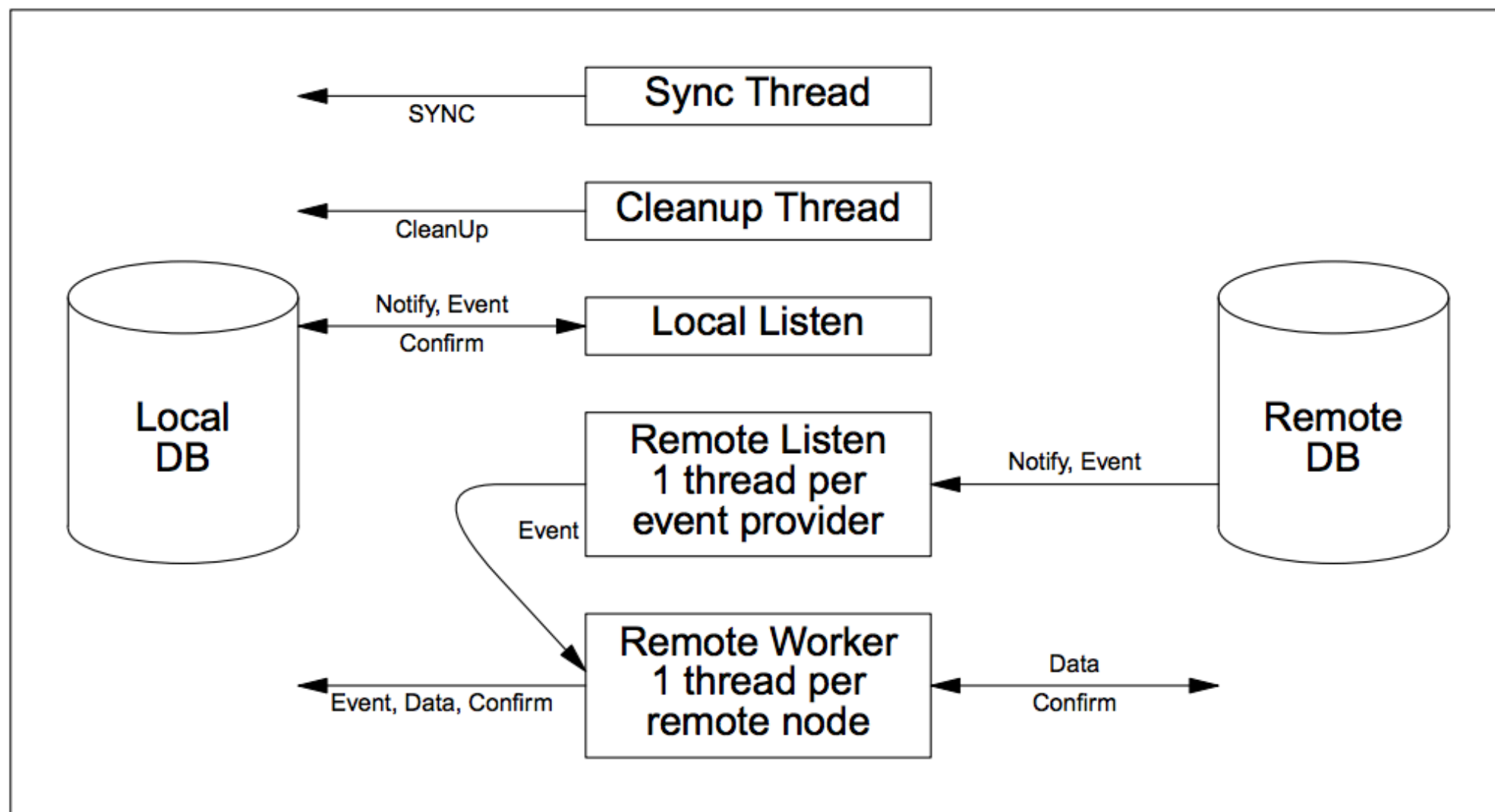


Figure 2

Slony

- ❖ <http://www.slony.info/images/Slony-I-concept.pdf>
- ❖ 2.4.1. Splitting the logdata

Slony — Splitting the logdata

- ❖ Splitting the logdata into groups of logically ascending transactions **is much easier** than someone might imagine.
- ❖ The Slony-I daemon will check in a configurable timeout if the log action sequence number of the local node has changed and if so, it will generate a SYNC event. All events generated by a system are generated in a serializable transaction and lock one object. It is thus guaranteed that their event sequence is the exact order in which they are generated and committed.
- ❖ An event contains among the message code and its payload information the entire serializable snapshot information of the transaction, that created this event. All transactions that committed between any two ascending SYNC events can thus be defined as

```
SELECT xid FROM logtable WHERE  
    (xid > sync1_maxxid OR (xid >= sync1_minxid AND xid IN (sync1_xip))) AND  
    (xid < sync2_minxid OR (xid <= sync2_maxxid AND xid NOT IN (sync2_xip)))  
;
```

- ❖ The daemon on the Slony-I local node only checks the local log action sequence, inserts a row and generates a notification if the sequence has changed.

Спасибо @mikhailtyurin

теперь мы знаем, что:

- ❖ на реплике нет SSI, ни на какой
- ❖ и кстати, что SSI в pg — не есть S2PL
- ❖ надо самим ловить аномалии расставляя блокировки строк/таблиц
- ❖ КОМЬЮНИТИ озадачено — думает

