

Effective development of Firebird database applications in Delphi/Lazarus

Dmitrii Kuzmenko, IBSurgeon

Firebird Conference 2019

Berlin, 17-19 October



YOUR PREMIER SOURCE OF FIREBIRD SUPPORT

IBSurgeon



**MOSCOW
EXCHANGE**



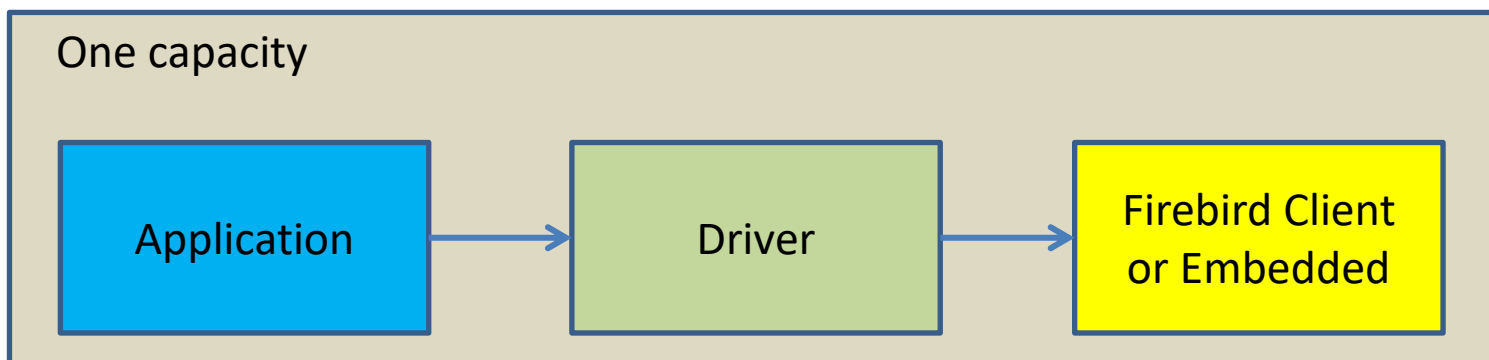
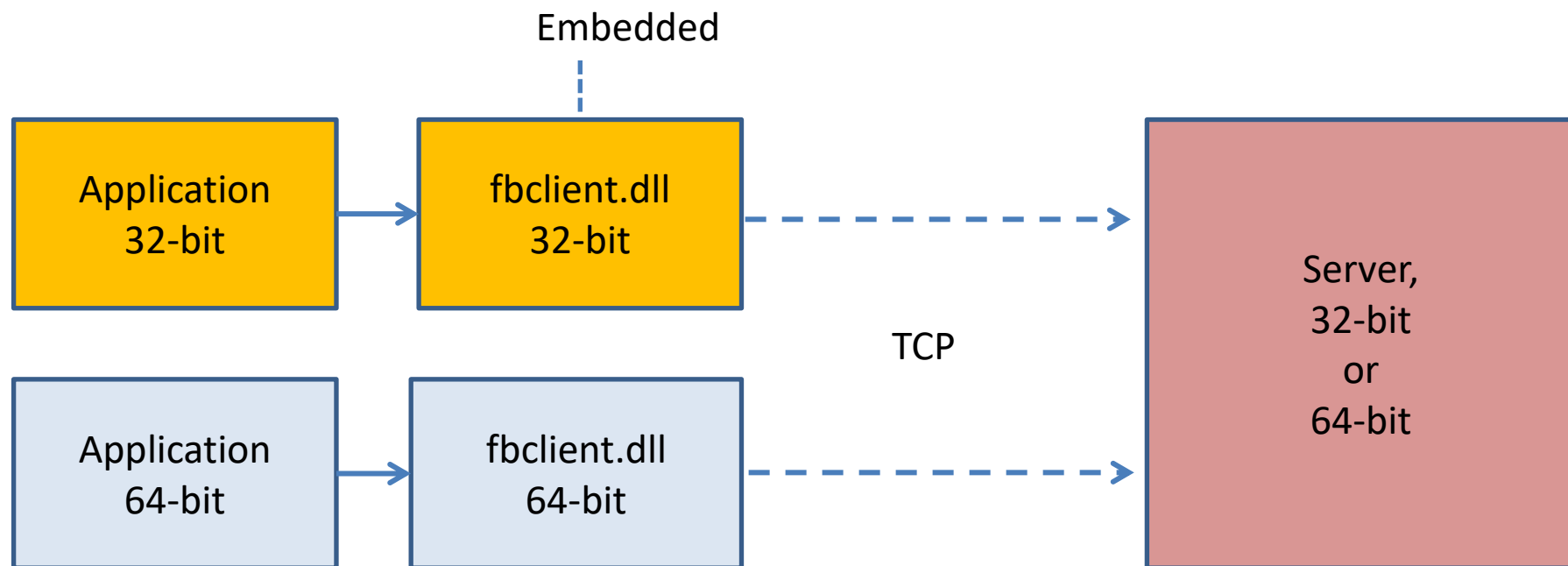
Fast Reports
Reporting must be fast!



List of topics

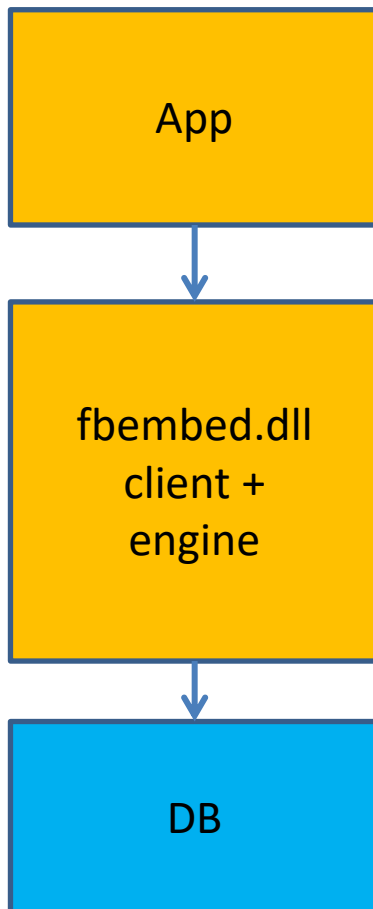
- Client Library
- Delphi, Lazarus
- Unicode
- Transactions
- Other
 - Reports
 - Stored Aggregates
 - Explicit locks
 - Data Editing

Client and server, 32 and 64 bit

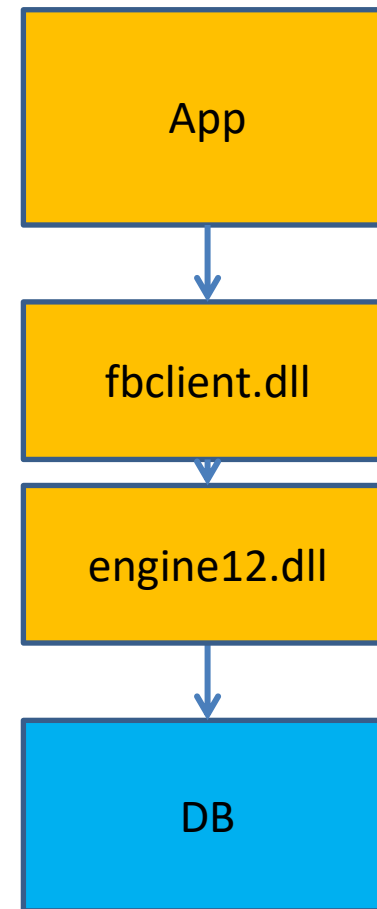


Exe + dll

1.5-2.5



3.0



Embedded - Development and usage

- Development is uncomfortable
 - ServerMode=SuperClassic, not SuperServer
 - Better use normal Firebird server installation
- 32bit or 64bit – to your taste
- Usage – single-user applications. Not multi-user
 - SuperClassic mode allows to work with one database for several applications on the same computer

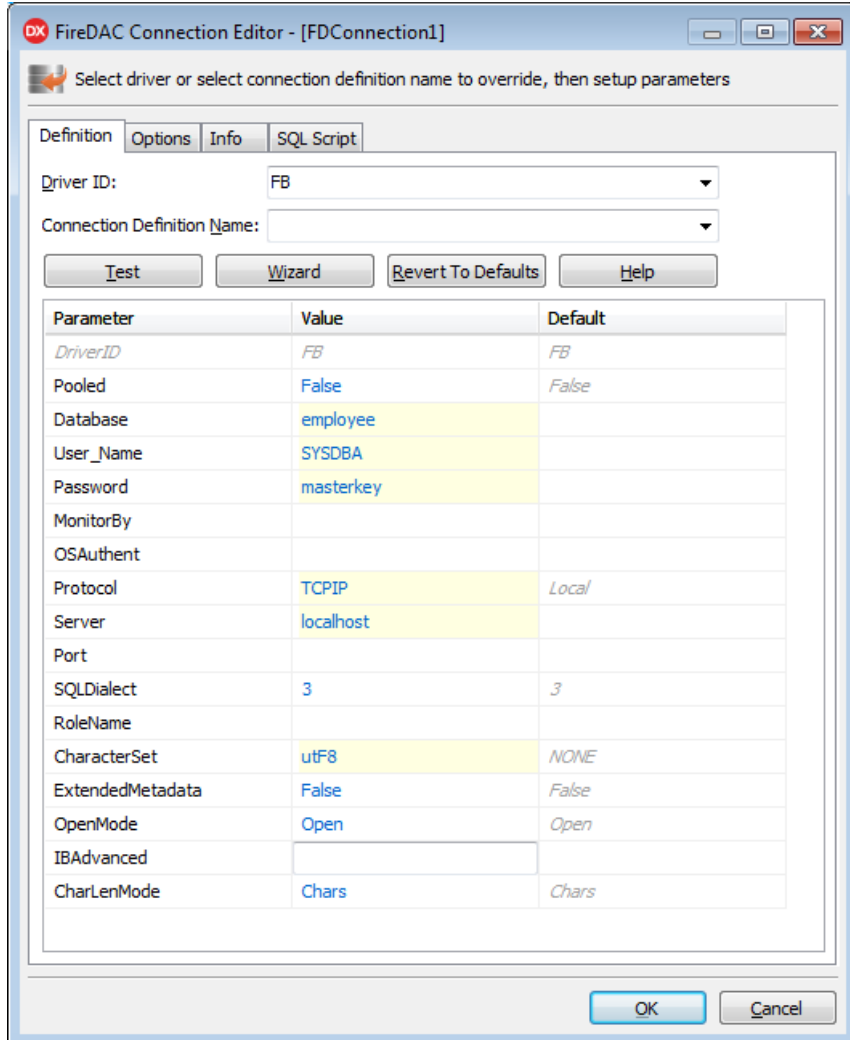
Delphi

- FireDAC
 - IBX
 - dbExpress
 - IBObjects
 - FIBPlus – discontinued
 - IBDAC, UNIDAC
 - ...
-
- in the beginning of the 2000 there were around 40 different drivers and component sets

Delphi

- Delphi – 32bit IDE
- Can compile 32bit and 64bit applications
- so, you need to have 32bit and 64bit fbclient.dll installed (by instclient i f)
- instclient i g - may also be needed for IBX (gds32.dll)

Delphi - FireDAC



FireDAC Connection Editor - [FDConnection1]

Select driver or select connection definition name to override, then setup parameters

Definition Options Info SQL Script

Driver ID: FB

Connection Definition Name:

Test Wizard Revert To Defaults Help

Parameter	Value	Default
DriverID	FB	FB
Pooled	False	False
Database	employee	
User_Name	SYSDBA	
Password	masterkey	
MonitorBy		
OSAuthent		
Protocol	TCP/IP	Local
Server	localhost	
Port		
SQLDialect	3	3
RoleName		
CharacterSet	utf8	NONE
ExtendedMetadata	False	False
OpenMode	Open	Open
IBAdvanced		
CharLenMode	Chars	Chars

OK Cancel

see Firebird documentation
Firebird 3.0 Developer's Guide
Developing Firebird Applications in Delphi

FDTransaction.Params

dbExpress – handles?

- Cannot set transaction parameters
- TDBXTransaction exists, but useless
- Cannot switch between transactions
- transaction1.BeginTransaction;
- ...
- transaction2.BeginTransaction;
- ... here you can not return to transaction1 context, you can only call it's commit/rollback.

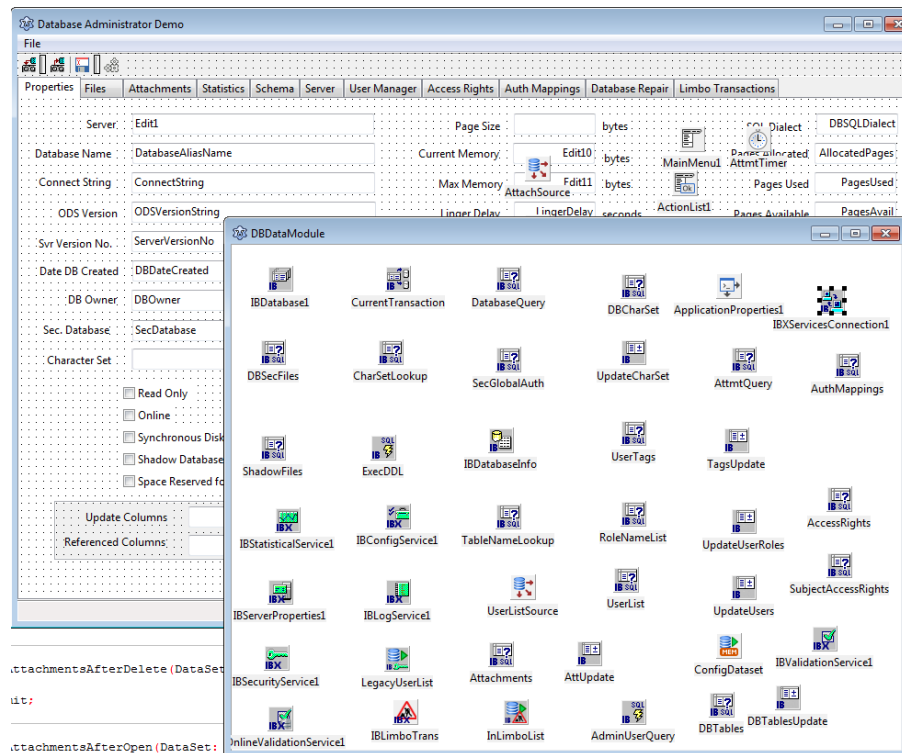
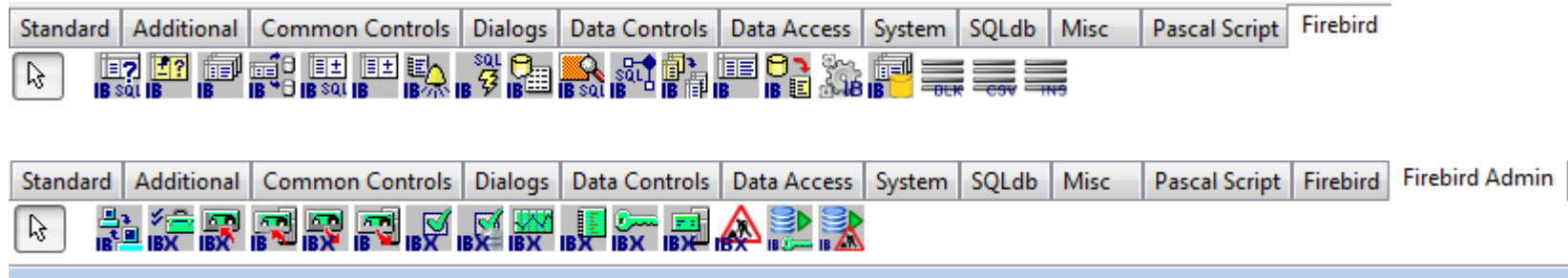
Lazarus

- Components
- IBX2
 - contains FB 3 API wrapper
- ZeosDBO
- FBLib

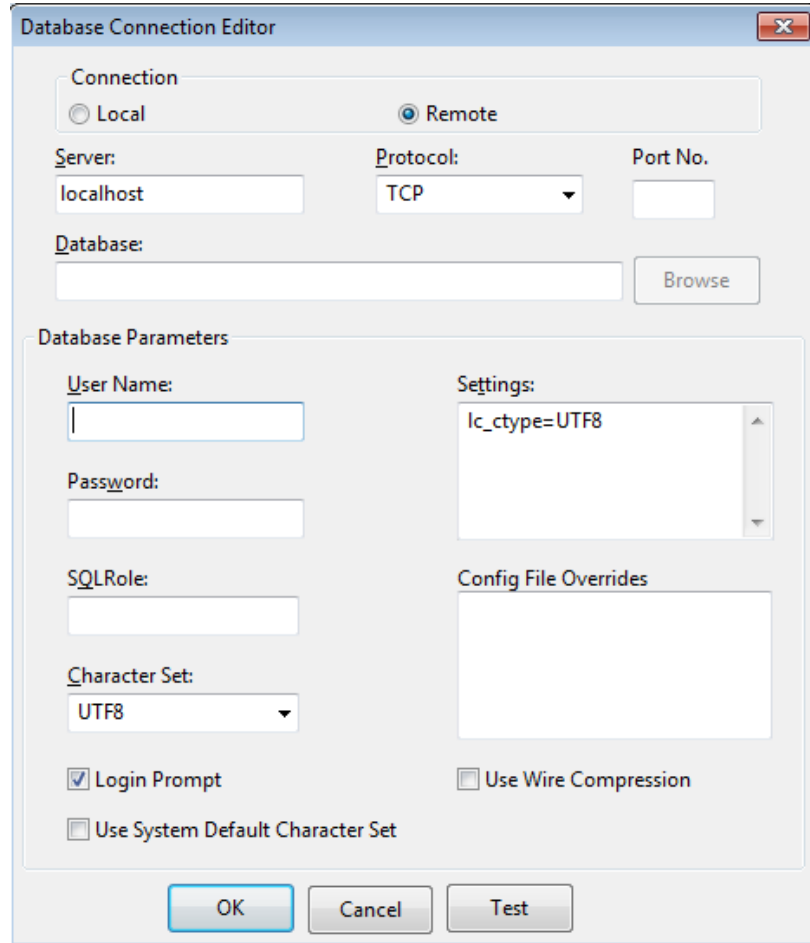
Client libraries

- Lazarus – 64bit IDE
- all components need 64bit fbclient.dll
- use instreg i f
- or put fbclient to the lazarus and application folder

Lazarus - IBX



IBX



The screenshot shows the 'Database Connection Editor' dialog box. It has a title bar with a close button. The 'Connection' section has two radio buttons: 'Local' and 'Remote', with 'Remote' selected. Below this are three fields: 'Server:' with the text 'localhost', 'Protocol:' with a dropdown menu showing 'TCP', and 'Port No.' which is empty. There is a 'Database:' field with a 'Browse' button next to it. The 'Database Parameters' section contains several fields: 'User Name:' (empty), 'Password:' (empty), 'SQLRole:' (empty), and 'Character Set:' with a dropdown menu showing 'UTF8'. There are also two checkboxes: 'Login Prompt' (checked) and 'Use System Default Character Set' (unchecked). To the right of these fields is a 'Settings:' list box containing 'lc_ctype=UTF8'. Below the settings is a 'Config File Overrides' text area. At the bottom are three buttons: 'OK', 'Cancel', and 'Test'.

Database Connection Editor

Connection

☐ Local ☒ Remote

Server: localhost Protocol: TCP Port No.

Database: Browse

Database Parameters

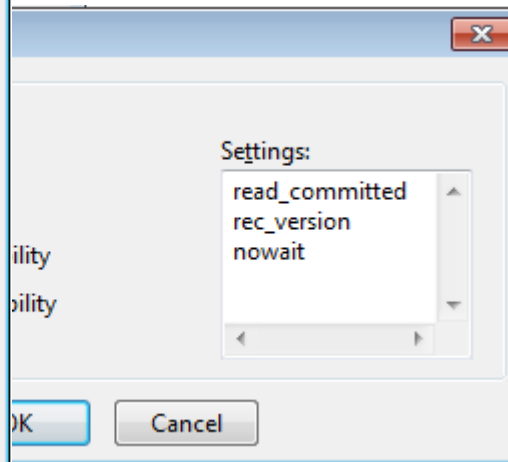
User Name: Password: SQLRole: Character Set: UTF8

Settings: lc_ctype=UTF8

Config File Overrides

☒ Login Prompt ☐ Use Wire Compression ☐ Use System Default Character Set

OK Cancel Test



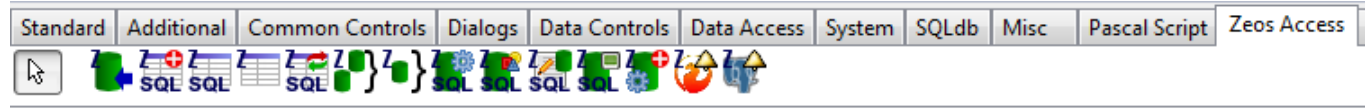
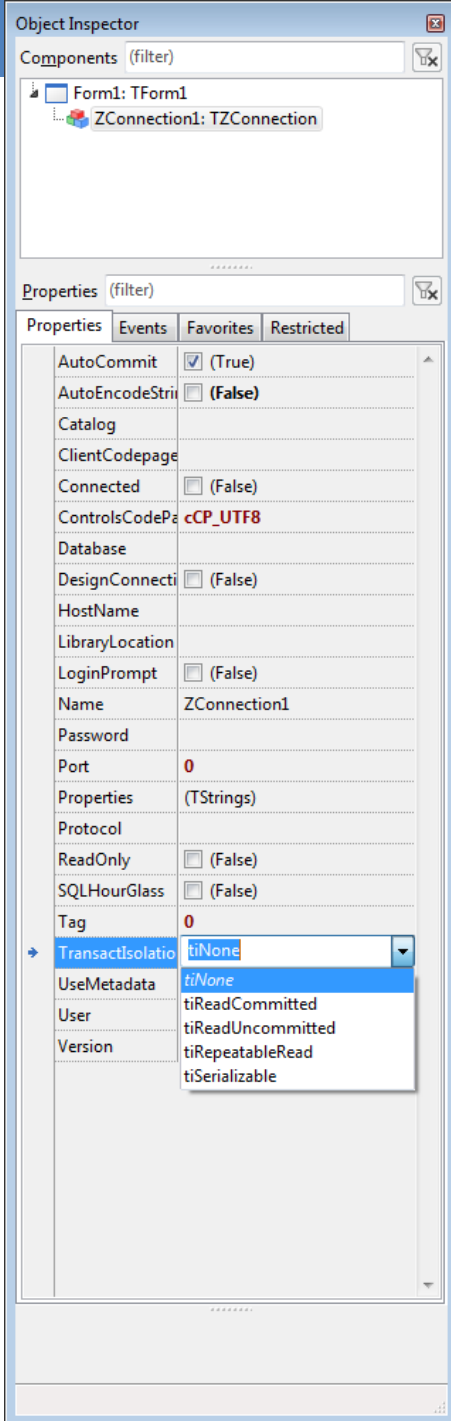
The screenshot shows a smaller dialog box, likely a sub-dialog of the main one. It has a title bar with a close button. It contains a 'Settings:' list box with the following items: 'read_committed', 'rec_version', and 'nowait'. At the bottom are two buttons: 'OK' and 'Cancel'.

Settings:

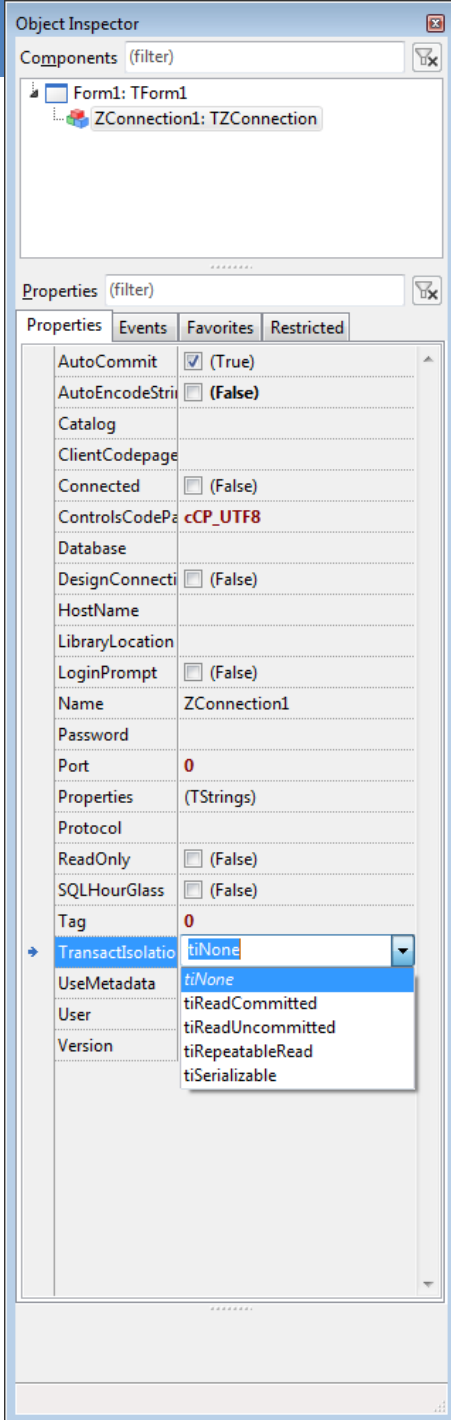
- read_committed
- rec_version
- nowait

OK Cancel

ZeosDBO



- No Transacton component
- if not IConnection.InTransaction then
 IConnection.BeginTransaction;
if IConnection.InTransaction then
 IConnection.CommitTransaction;
- Default to BDE – autocommit,
etc...



- ```
ZConnection.TransactIsolationLevel := tiNone;
ZConnection.Properties.Add('isc_tpb_concurrency');
ZConnection.Properties.Add('isc_tpb_wait');
ZConnection.Properties.Add ('lc_ctype=ISO8859_1');
ZConnection.Connect;
```

# Working with components

- Table component – only for reference tables
- “Live dataset” does not exist
  - Select, insert, update, delete, refreshRow
- Main problem – invalidation of a cursor after commit/rollback
  - you may use ClientDataSet (briefcase model)

# Components summary

- Better use components with transaction control, with ability to set Firebird specific connection and transaction parameters, and with Services API support
- Universal components (including ODBC) – for universal (different SQL servers) development

# Unicode

- Delphi up to 2007
  - non-unicode
- Delphi 2009 and higher
  - unicode, UTF-16, UTF-8
- Lazarus
  - unicode, UTF-8

# Unicode

- database in country charset
  - ISO8859\_1, WIN1252 – Western Europe
    - Danmark, Germany, Netherlands, UK, Spain, Finland, France, Iceland, Italy, Norway, Portugal, Brasil, Sweden
  - ISO8859\_2, WIN1250 – Central Europe
    - Czech, Hungary, Poland
  - ISO8859\_4 – Northern Europe
    - Estonia, Latvia, Lithuania, Greenland
- database in unicode (UTF8)

# Charsets

- **Connection charset** – only one particular charset
- **Database charset** – each character or text blob column can have it's own charset
  - ! “database charset” is a default charset that will be used when new character/blob column is created, and no charset is specified. In this sense database has no “charset”.*
  - rdb\$database.rdb\$character\_set\_name*
  - rdb\$character\_sets.rdb\$default\_collate\_name* – from 2.5
- Example:
  - if connection charset is ISO8859\_1, and database have character columns of ISO8859\_2, there must be “conversion table” from the column charset to connection charset

# Is unicode is bigger?

## Test:

X1251 varchar(30) character set win1251

XUTF8 varchar(30) character set UTF8

100k records.

Win1251 is a single-byte character set. So, 30 characters = 30 bytes

UTF8 is dynamic character set. Russian characters here occupy 2 bytes per each, latin characters are 1 byte per character.

So, let us fill data with the national (russian) characters only.

| Table | Records | RecLength    | Data Pages  | Size, mb    |
|-------|---------|--------------|-------------|-------------|
| X1251 | 100000  | <b>28.86</b> | <b>852</b>  | <b>6.66</b> |
| XUTF8 | 100000  | <b>49.01</b> | <b>1094</b> | <b>8.55</b> |

UTF8 storage takes ~30% more than single-byte character set

# To and back

- Connection win1251 – database win1251
- Connection utf8 – database win1251
  - only win1251 characters will pass
- Connection win1251 – database utf8
  - only win1251 characters will pass
- As a transition – first use UTF8 connection charset, next upgrade your database

# How to upgrade to UTF8?

- Only by pumping data to the new UTF8 DB.
  - take script from the db (`isql -x -ch nnn`)
    - -ch is mandatory. otherwise you may get garbage instead of literals and comments
  - check any specification of “character set”
  - check any “collation” specification, decide what to specify
    - UTF8 (USC\_BASIC), UNICODE, UNICODE\_CI, UNICODE\_CI\_AI
  - check column sizes that are close to 32k bytes ( $32k/4=8k$ )
  - create database
  - pump data from original database to the new one  
(don't worry to use non-unicode pump, just use same character set for both connections, data will be converted to unicode automatically)

# Upper

- Old style
  - select \* from table  
where upper(name) = 'STRING'
- New style
  - declare column as  
name varchar(30) collate unicode\_ci
  - select \* from table  
where name = 'string'

# iOS, Android?

- Only UTF8 databases.

# Working with transactions

- Worst example – MastApp.  
One transaction for everything  
CommitRetaining
- Do not use CommitRetaining (or RollbackRetaining)
- Use as many transactions components, as you need
- Do not use IBTransaction.Active:=True/False
  - use StartTransaction, Commit and Rollback methods.  
Active:=False is equal to Rollback (by default)
- Do not start transactions too often
- Avoid long running transactions

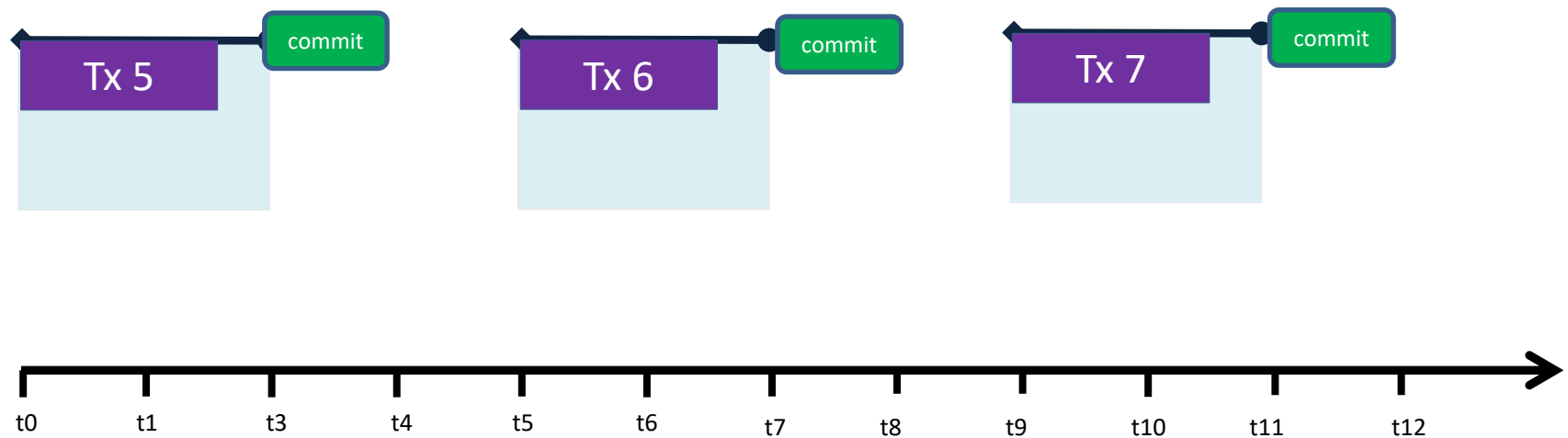
# Transaction defaults

- IBX – write wait snapshot
- FIBPlus – write no wait read\_committed  
rec\_version
- FireDAC - write no wait read\_committed
- Others - ? check by yourself

# Main performance issues

- Transaction control
  - garbage, slowness
- Query optimization
  - bad SQL, no index, redundant index
- firebird.conf optimization
- Hardware optimization

# Sequential transactions

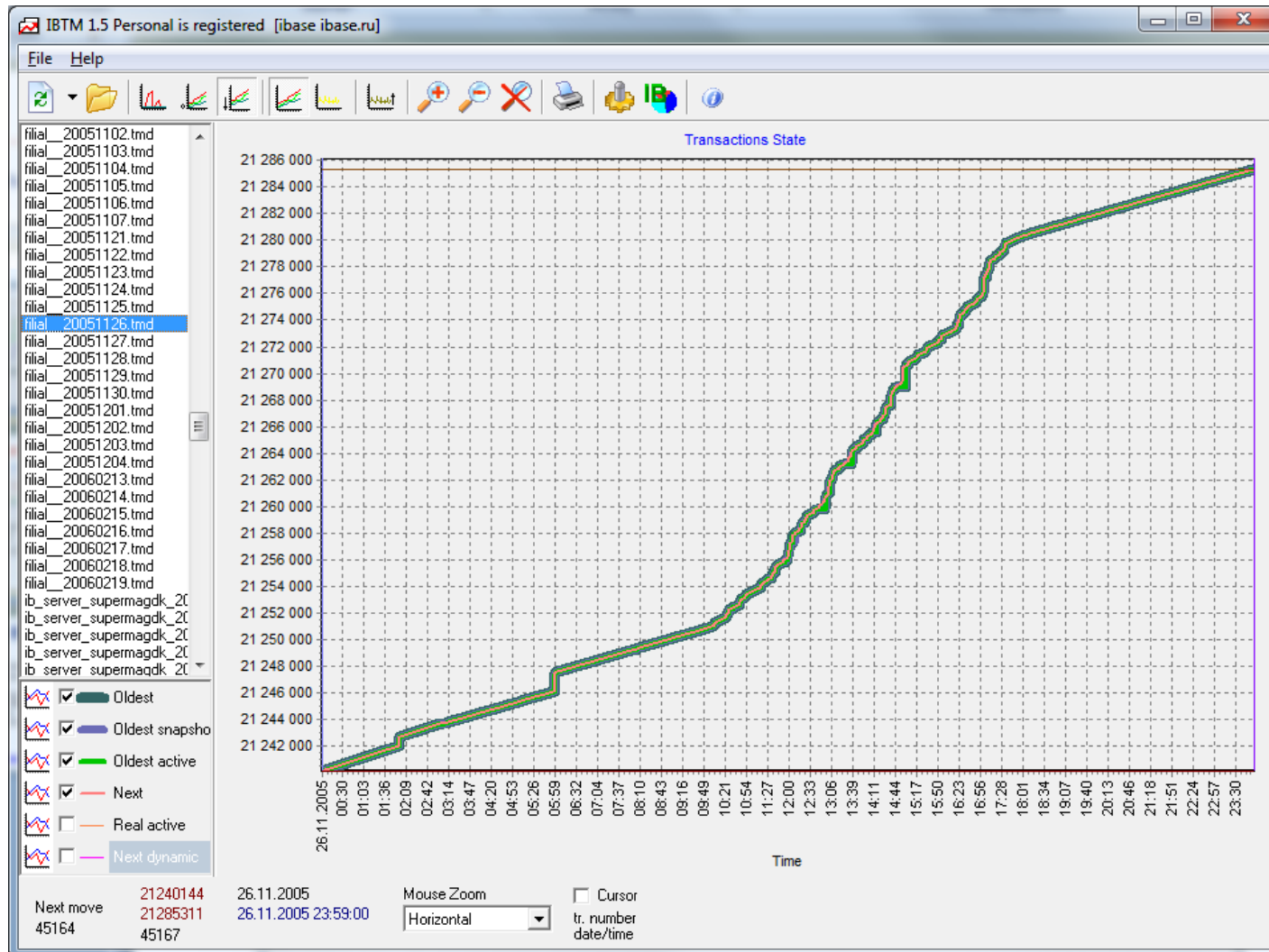


Always test your application in the multi-user mode

# Gstat -h

- Database header page information:
  - Flags 0
  - Checksum 12345
  - Generation 112431494
  - Page size 8192
  - ODS version 11.1
  - **Oldest transaction 100 x-1**
  - **Oldest active 101 x**
  - **Oldest snapshot 101 x**
  - **Next transaction 102 x+1**
  - Bumped transaction 1
  - Sequence number 0
  - Next attachment ID 0
  - Implementation ID 16
  - Shadow count 0
  - Page buffers 0
  - Next header page 0
  - Database dialect 1
  - Creation date Jun 5, 2011 10:02:19
  - Attributes force write
- Variable header data:
  - Sweep interval: 20000
  - \*END\*

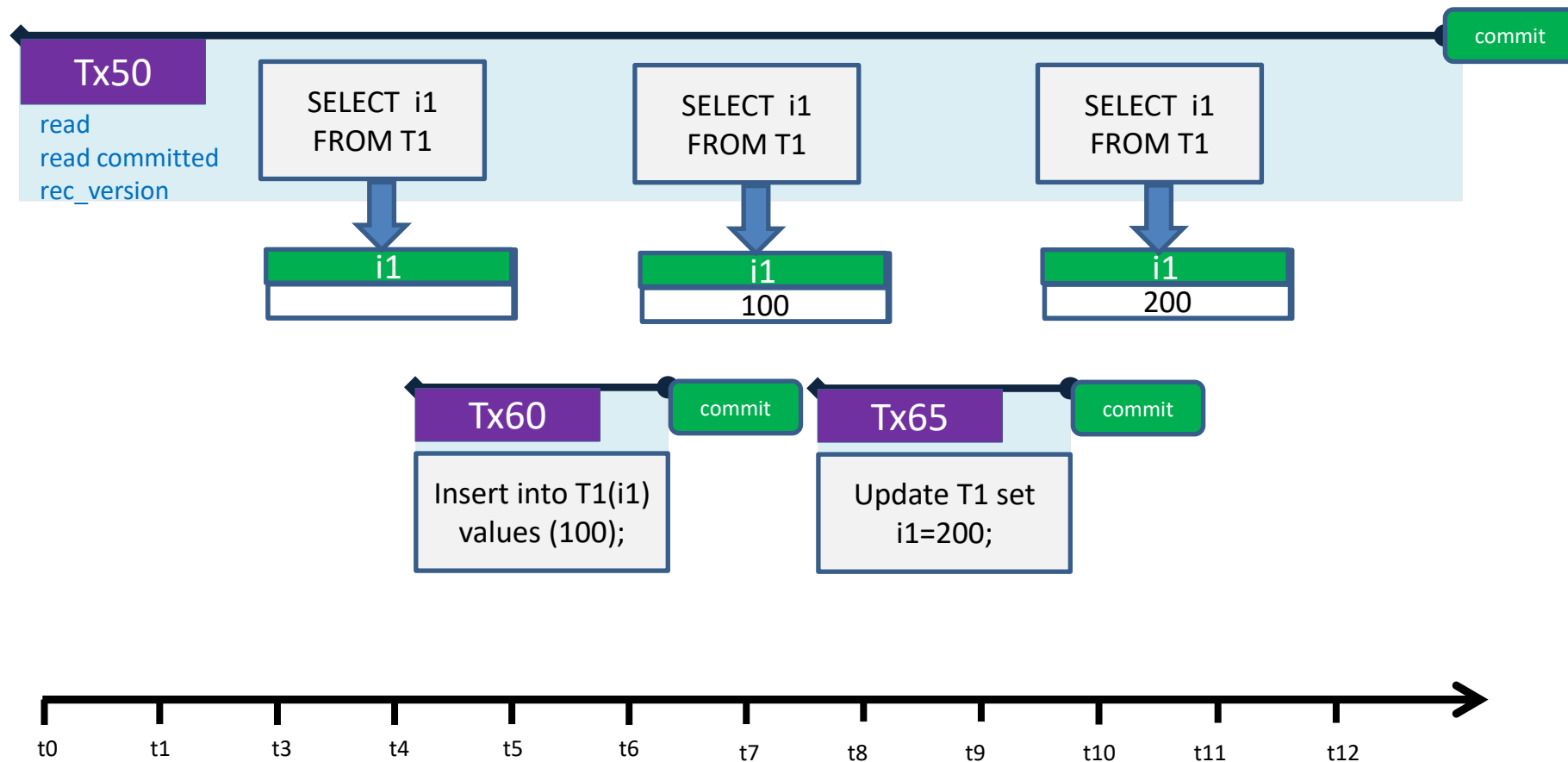
# Ideal transaction control



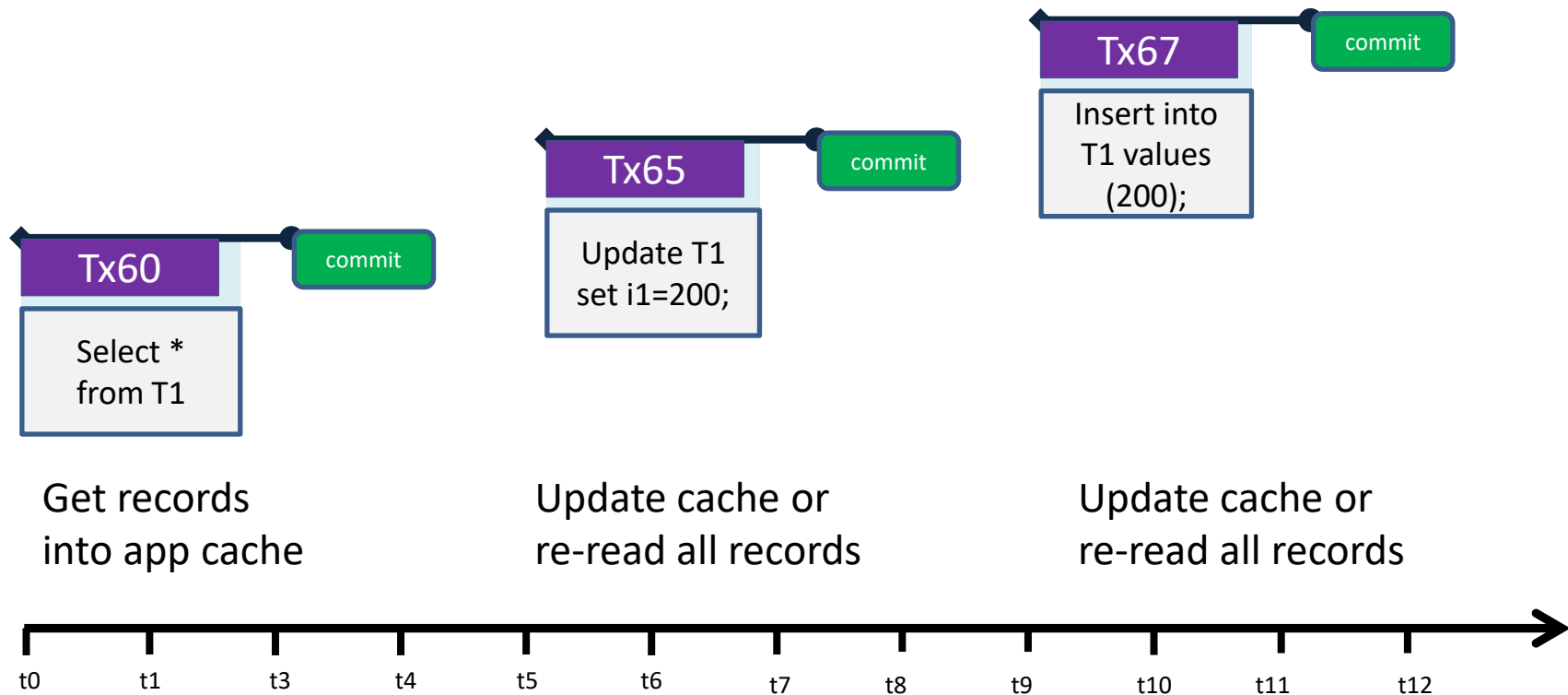
# Two ways to almost ideal transaction control

1. Long read-only read-committed (until Firebird 4.0) and short write
2. Short read and write

# Long read-only RC and short write



# Short read and write



# Pro & Contra

## **Long read-only RC and short writes**

- + easy to implement read and update logic
- - requires support from drivers/components (2 transactions or 2 connections)
- + more convenient for client-server
- - less convenient for multi-tier and stateless applications

## **Short read and writes**

- - hard to implement sophisticated caching
- + works with any data access drivers/components
- - less convenient for client-server
- + more convenient for multi-tier and stateless applications

**WHY NOTHING IS PERFECT**

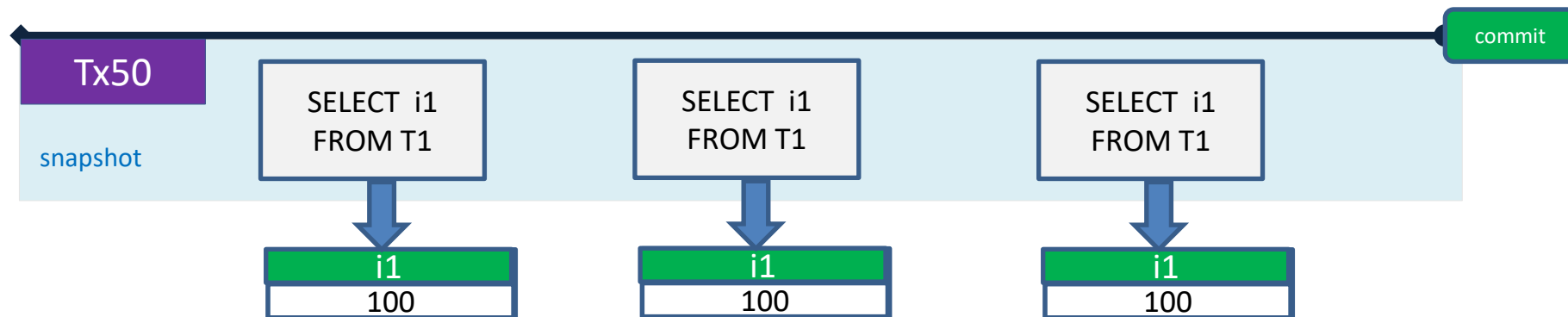
# Exceptions from ideal transaction control

- Reports
- Product balances
- Explicit record locking
- Data editing
- Robots

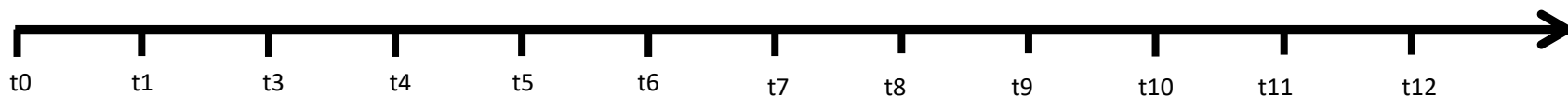
# Reports

- Need data consistency
- Long queries
- Complex reports read the same data several times

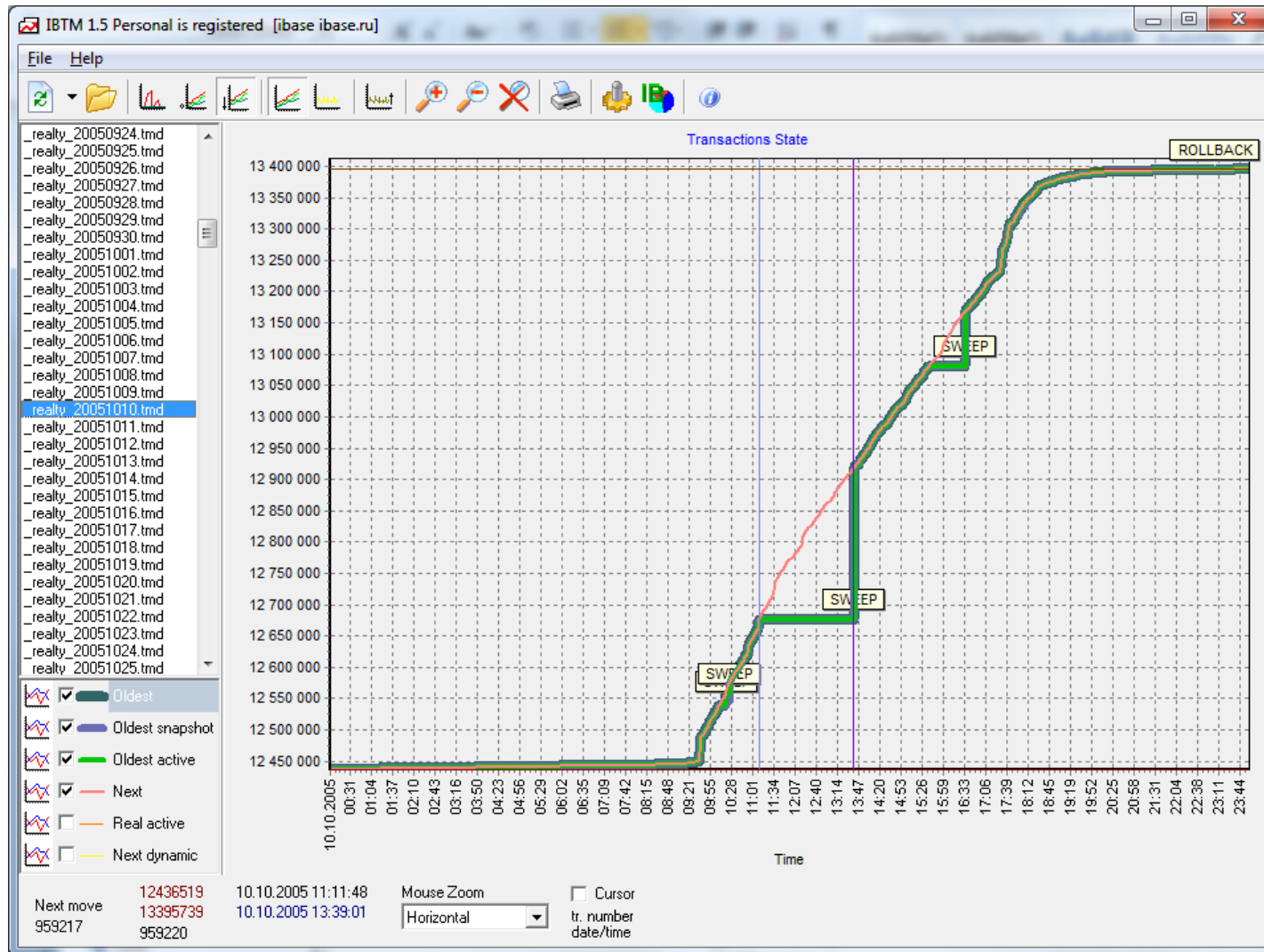
# Reports - snapshot



No difference – wait/nowait (except concurrency), read/write



# Heavy report example – OAT stuck



# How to workaround long report problem

- Most reports does not need real-time data
- Change logic of data processing
- Scaling
  - Replication
  - Transferring data to another DB with Execute Statement On External
  - Nbackup

# Change logic of data processing – Stored aggregates

- 1 order - ~10 goods
- 100 orders per day
- $100 * 10 = 1000$  records per day
- 365000 records per year
  
- Store “order\_total” in ORDERS table – 10 times less records
- **Pro**: less records, faster queries
  - No update conflicts if there is no concurrent order editing
- **Con**: additional field in ORDERS

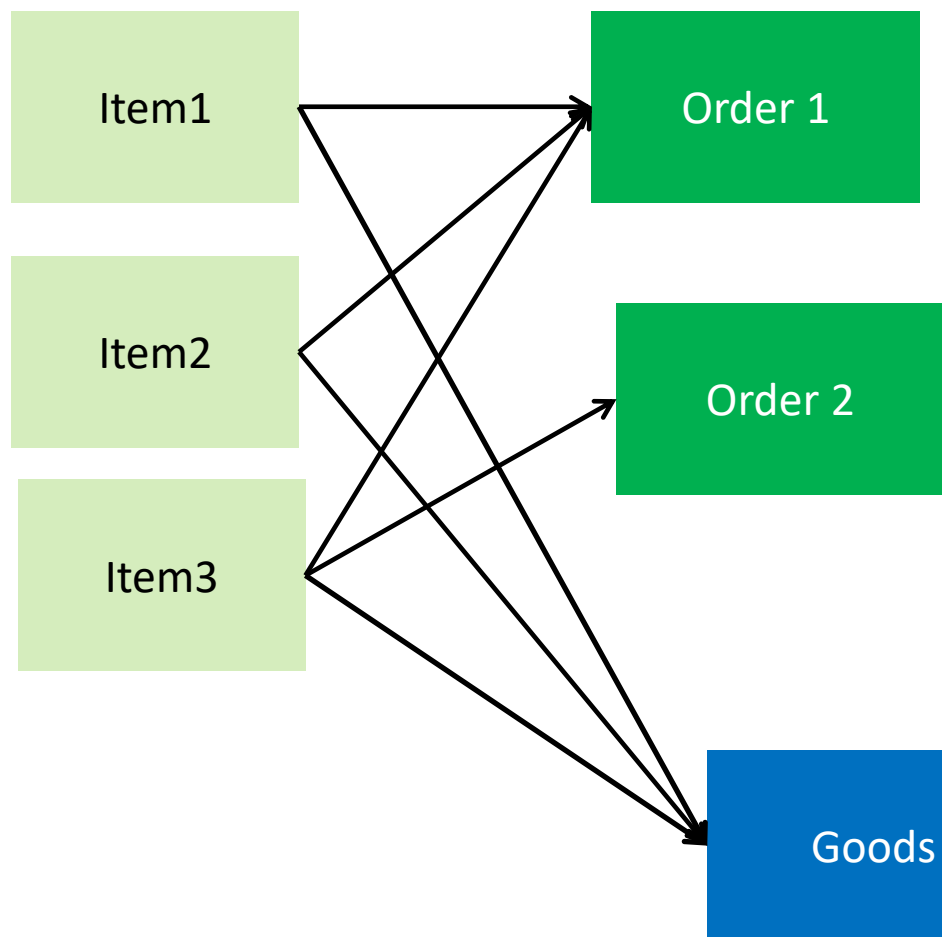
# If you want to go further...

- To store sum by day, month, ...
- Updates by triggers “in place” won’t work – too high possibility of lock conflicts
- Solution? Regular updates
  - Regular procedure must be run in exclusive mode
    - Using generator
    - Using consistency isolation mode
  - By schedule (at night)

# Goods balances – update locks

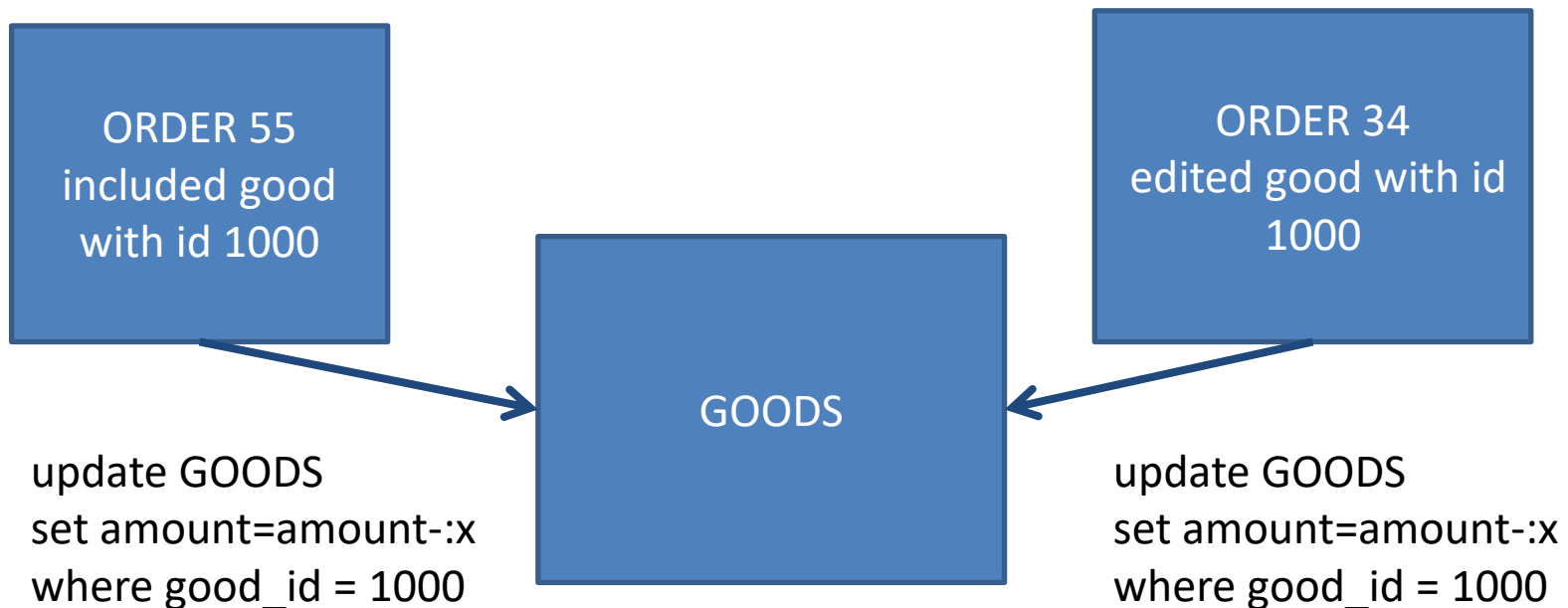
- Change AMOUNT while order is processed
- 
- Insert - set  $AMOUNT = AMOUNT - new.INORDER$
- Delete - set  $AMOUNT = AMOUNT + new.INORDER$
- Update - set  $AMOUNT = AMOUNT + new.INORDER - old.INORDER$
- There may be conflicts when 2 people sell same `good_id`
  - Long transaction will lock all concurrent order processing
  - Short transactions have less chances to get update conflict, and may be retried

# Update locks



ok for the order, while only 1 user  
may edit 1 order

not ok for goods, since  
same item may be sold in  
2 (or more) orders, that may  
be edited same time



**Editing one order by 2 users is a rare case, but using same item is not rare**

# Goods balance - solution

- CREATE TABLE MOVEMENTS(  
GOOD INTEGER NOT NULL REFERENCES GOODS,  
AMOUNT INTEGER NOT NULL)
- CREATE TABLE GOODS\_AMOUNTS\_AGG(  
GOOD INTEGER NOT NULL REFERENCES GOODS,  
AMOUNT INTEGER NOT NULL)
- On insert delete and update MOVEMENTS do
- **INSERT** INTO GOODS\_AMOUNT\_AGG  
(GOOD, AMOUNT) VALUES
  - (NEW.GOOD, NEW.AMOUNT);
  - (NEW.GOOD, NEW.AMOUNT-OLD.AMOUNT);
  - (OLD.GOOD, -OLD.AMOUNT);

- CREATE VIEW GOODS\_AMOUNT  
(GOOD, AMOUNT) AS  
    SELECT GOOD, SUM(AMOUNT)  
    FROM GOODS\_AMOUNT\_AGG  
    GROUP BY GOOD
- CREATE PROCEDURE GOODS\_AMOUNT\_ROLL\_UP AS  
    DECLARE GOOD INTEGER;  
    DECLARE TOTAL INTEGER;  
    BEGIN  
        FOR SELECT GOOD, SUM(AMOUNT)  
        FROM GOODS\_AMOUNT\_AGG  
        GROUP BY GOOD  
        HAVING COUNT(\*) > 1 – *interested of 2 or more records*  
        INTO :GOOD, :TOTAL  
        DO  
            BEGIN  
                DELETE FROM GOODS\_AMOUNT\_AGG  
                WHERE GOOD=:GOOD;  
                INSERT INTO GOODS\_AMOUNT\_AGG  
                (GOOD, AMOUNT) VALUES(:GOOD, :TOTAL);  
            END  
    END
- Run procedure in concurrency (or consistency)

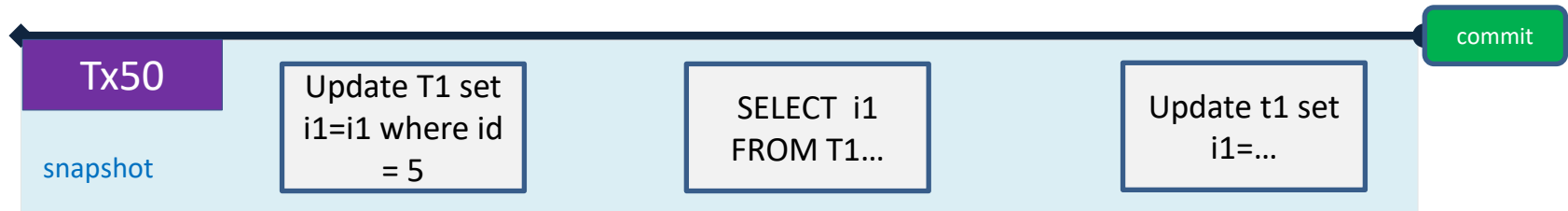
# Exclusive document editing

- Goal – implement exclusive changes
- Rollbacks are not welcome
- Need explicit record locking

# How to implement explicit record locking

- Blank update in long transaction
  - Or `SELECT ... FOR UPDATE WITH LOCK`
- Flags at business logic level

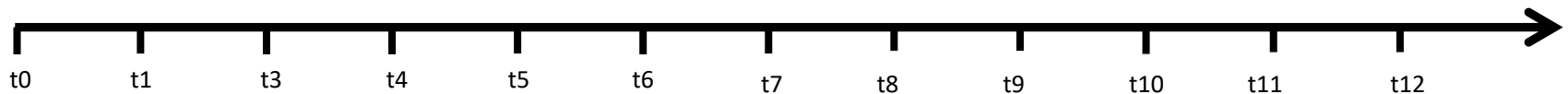
# Blank update



First update creates record version,  
preventing other transactions to update this record

Pro: Easy to implement

Con: Trigger fires on update, 2<sup>nd</sup> update causes update “in-place”



# SELECT ... FOR UPDATE WITH LOCK

- Same as blank update
- Can lock several records
- Locks record on fetch
  - Result returns one record per one fetch (no buffering)
- Useless for aggregates (SUM, AVG, COUNT, ...)

- Locking in the versioning server is not normal
- It maybe not enough to choose appropriate transaction isolation level

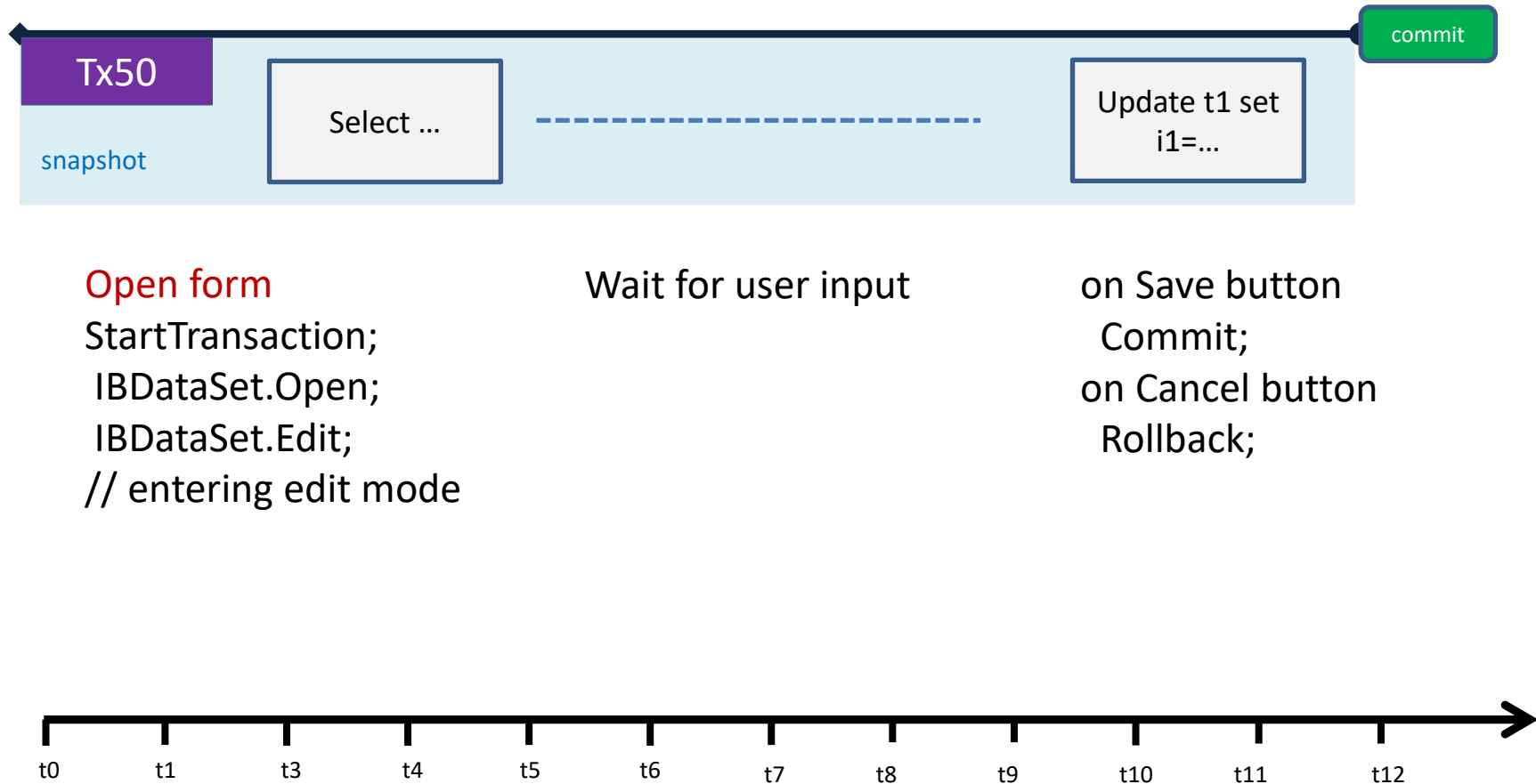
# Flags at business logic level

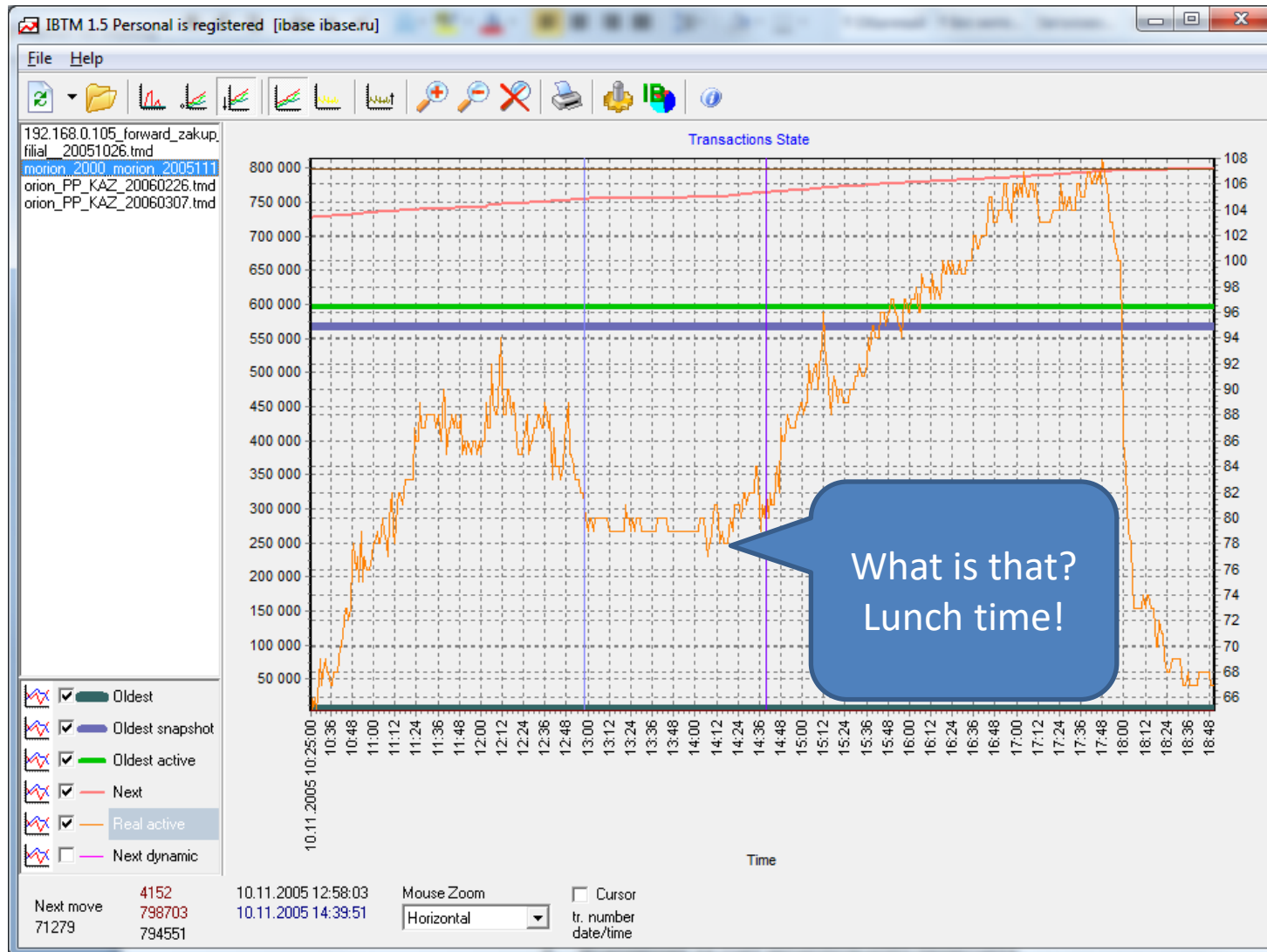
- Add User and TimeStamp fields, or create additional table
- When you want to “lock”, write USER and CURRENT\_TIMESTAMP in short transaction
- if user <> myself then
  - if TimeStamp is far then
    - UPDATE set User, TimeStamp
  - else Fail(“locked by user User at TimeStamp”)
- else
  - UPDATE set TimeStamp
- Additional table need to be cleared (disconnected apps)

# Data editing

- Application is used by operator not in the way developer designed it
- Badly designed data editing can be a problem

# Data editing: wrong scenario

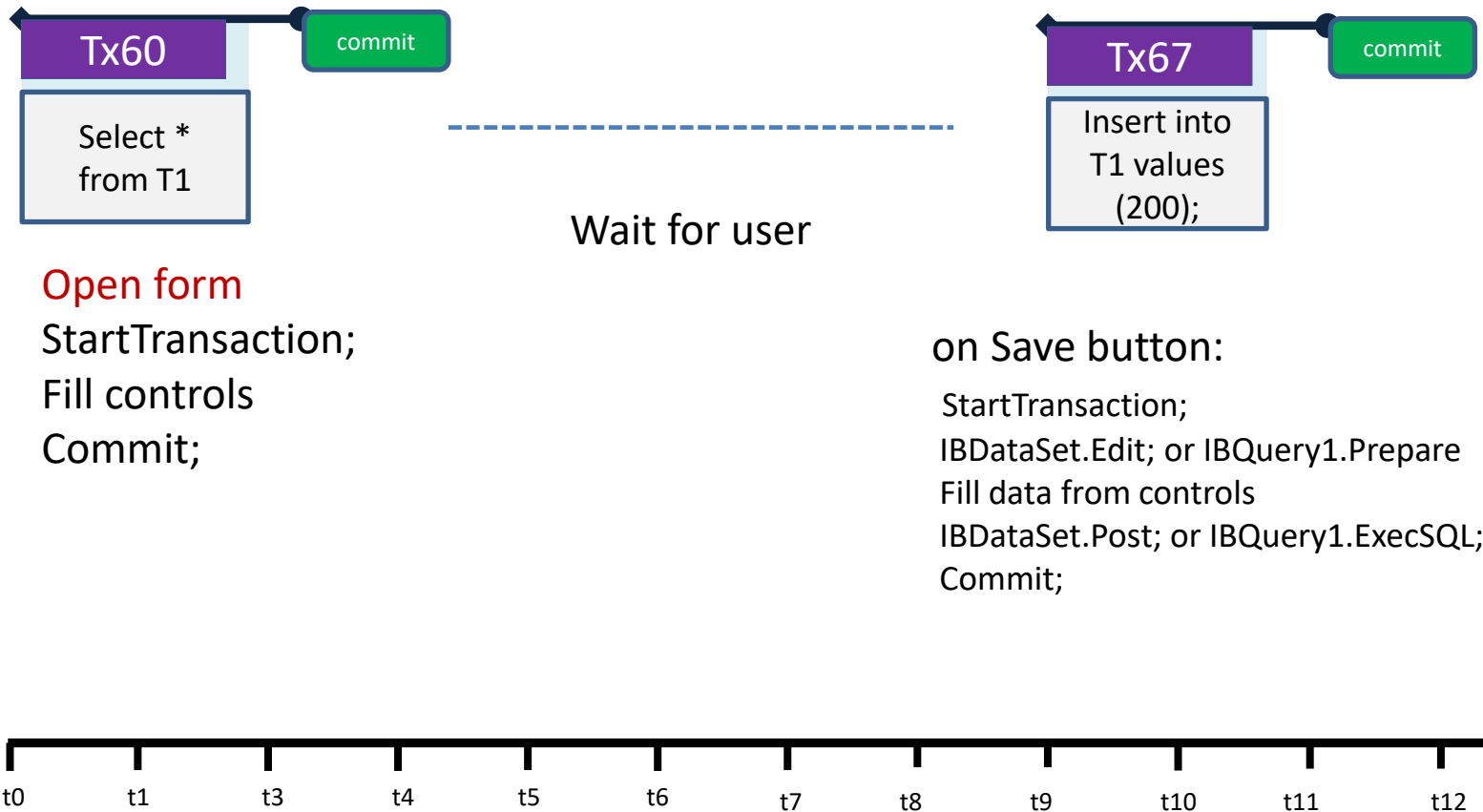




# Data editing: Solution

- Open form
- StartTransaction;
- Fill controls
- Commit;
- Wait for user
- User presses Save button:
- StartTransaction;
  - IBDataSet.Edit; or IBQuery1.Prepare
- Fill data from controls
  - IBDataSet.Post; or IBQuery1.ExecSQL;
- Commit;

# Data editing: Solution



# Robot rules

## Reading robots

- Use read-only  
ReadCommitted
- Try to do work in one transaction, if possible
- Multi-tier - connection and transaction pooling
- Goals
  - Do not stuck OAT
  - Do not advance Next too much

## Writing robots

- Do not keep attachment open
  - attach, do work, close;
- Keep transactions short
- Try to do work in one transaction, if possible
- Goals
  - Do not stuck OAT

# Tools to monitor transaction markers

- `gstat -h`
- `mon$transactions` (Firebird 2.1 and higher)
- Trace (Firebird 2.5 and higher)
- HQBird DataGuard
- FBScanner
- IBAnalyst (`gstat` visual)

# About IBSurgeon

# IBSurgeon



- Tools and consulting
- Platinum Sponsor of Firebird Foundation
- Founded in 2002: 17 years of Firebird and InterBase recoveries and consulting
- Based in Moscow, Russia

[www.ib-aid.com](http://www.ib-aid.com)

[www.ibsurgeon.com](http://www.ibsurgeon.com)