Firebird Tour 2017
Advanced Trace API

Vlad Khorsun, Firebird Project

About Firebird Tour 2017

- Firebird Tour 2017 is organized by Firebird Project, IBSurgeon and IBPhoenix, and devoted to Firebird Performance.
- The Platinum sponsor is Moscow Exchange
- Tour's locations and dates:
  - October 3, 2017 – Prague, Czech Republic
  - October 5, 2017 – Bad Sassendorf, Germany
  - November 3, 2017 – Moscow, Russia

MOSCOW EXCHANGE

- Platinum Sponsor
- Sponsor of
  - «Firebird 2.5 SQL Language Reference»
  - «Firebird 3.0 SQL Language Reference»
  - «Firebird 3.0 Developer Guide»
  - «Firebird 3.0 Operations Guide»
- Sponsor of Firebird 2017 Tour seminars
- www.moex.com

PLATINUM

IBSurgeon

HQBird

Advanced Firebird for Big Databases

- Replication, Recovery and Optimization for Firebird since 2002
- Platinum Sponsor of Firebird Foundation
- Based in Moscow, Russia

www.ib-aid.com

Trace Session Configuration

- fbtrace.conf file in Firebird root directory
  - Self-documented
  - Show all possible trace parameters
  - Uses own XML-like syntax in Firebird 2.5
    - Different from firebird.conf, fbintl.conf, etc
  - In Firebird 3 all configuration files have same syntax
    - Syntax for trace configuration slightly changed

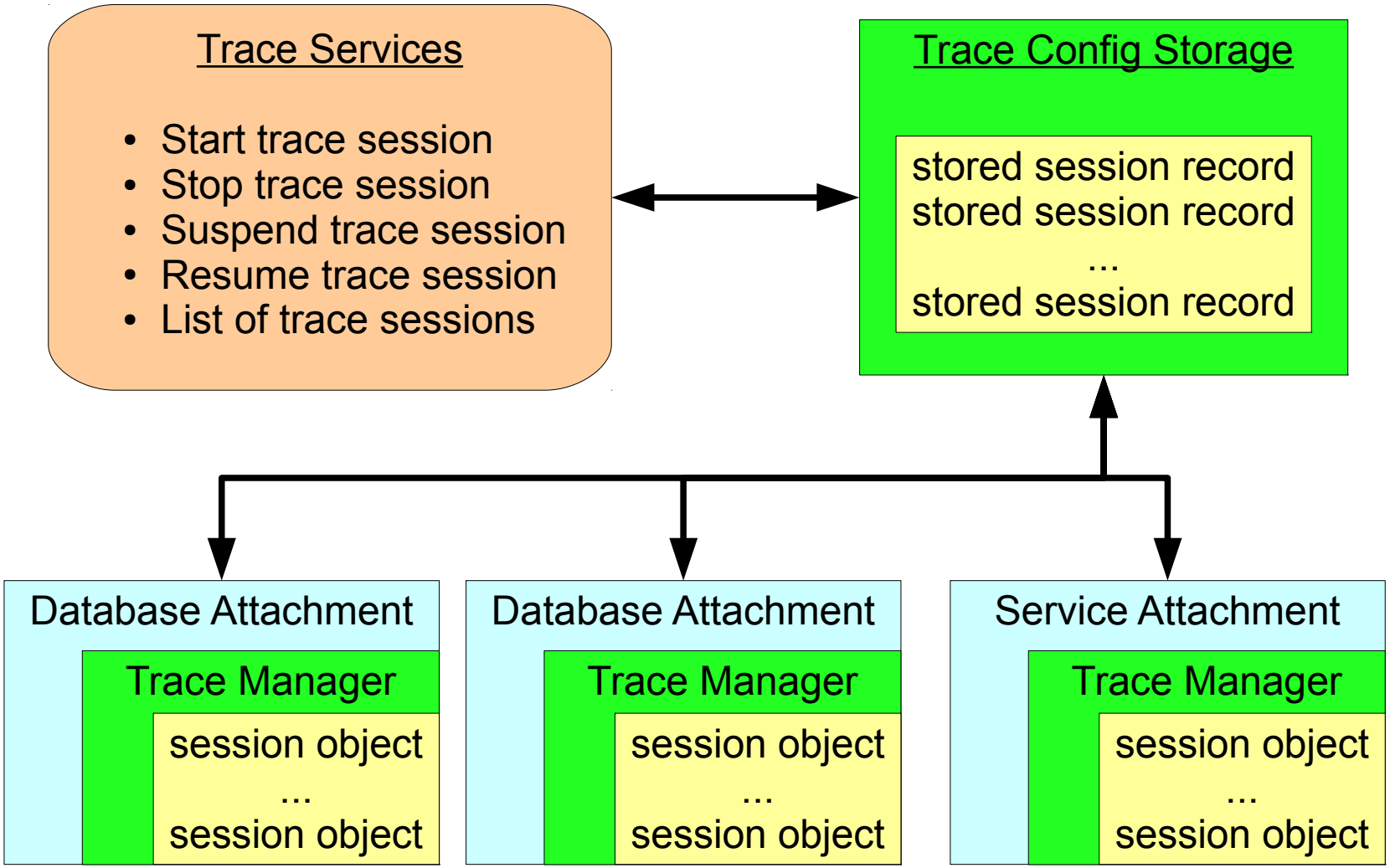
User Trace and System Audit

- User trace session
  - Initiated (started) by user via special service
  - Not preserved after Firebird shutdown
  - Output read by initiated service connection
  - Scope depends on user privileges
- Audit trace session
  - Initiated only by Firebird itself
  - Started with Firebird every time
  - Output stored in log file(s)
  - Scope is not limited

User Trace and System Audit

- User trace session
  - Could be temporary paused by the Firebird
  - Could be many user trace sessions
  - Could be managed by creator user or by SYSDBA
- Audit trace session
  - Never interrupted by Firebird
  - Only one audit trace session could exists
  - Could be managed by SYSDBA only

Trace Sessions in the Engine



Trace Sessions in the Engine

- To support multiply trace sessions we need some kind of manager
- Trace sessions need a place to keep its parameters
  - ID, name, creator user, state, etc
  - Trace session configuration

- TraceConfigStorage object is used for this goals
  - One instance per engine process
  - Common part is in shared memory
  - Changes increments special counter
  - Requires synchronization between engine processes

Trace Configuration Storage

- Consists from two files
  - *fb_trace*, control file, mapped into shared memory
  - *fb_trace_AAAAAA*, storage of trace sessions records
- Both files placed at Firebird lock directory
  - by default COMMON_APPDATA\firebird
- Creates when Firebird process starts
- Shared by all Firebird processes (embedded too !)
- Deleted when last Firebird process gone
  - trace sessions is not preserved between Firebird restarts

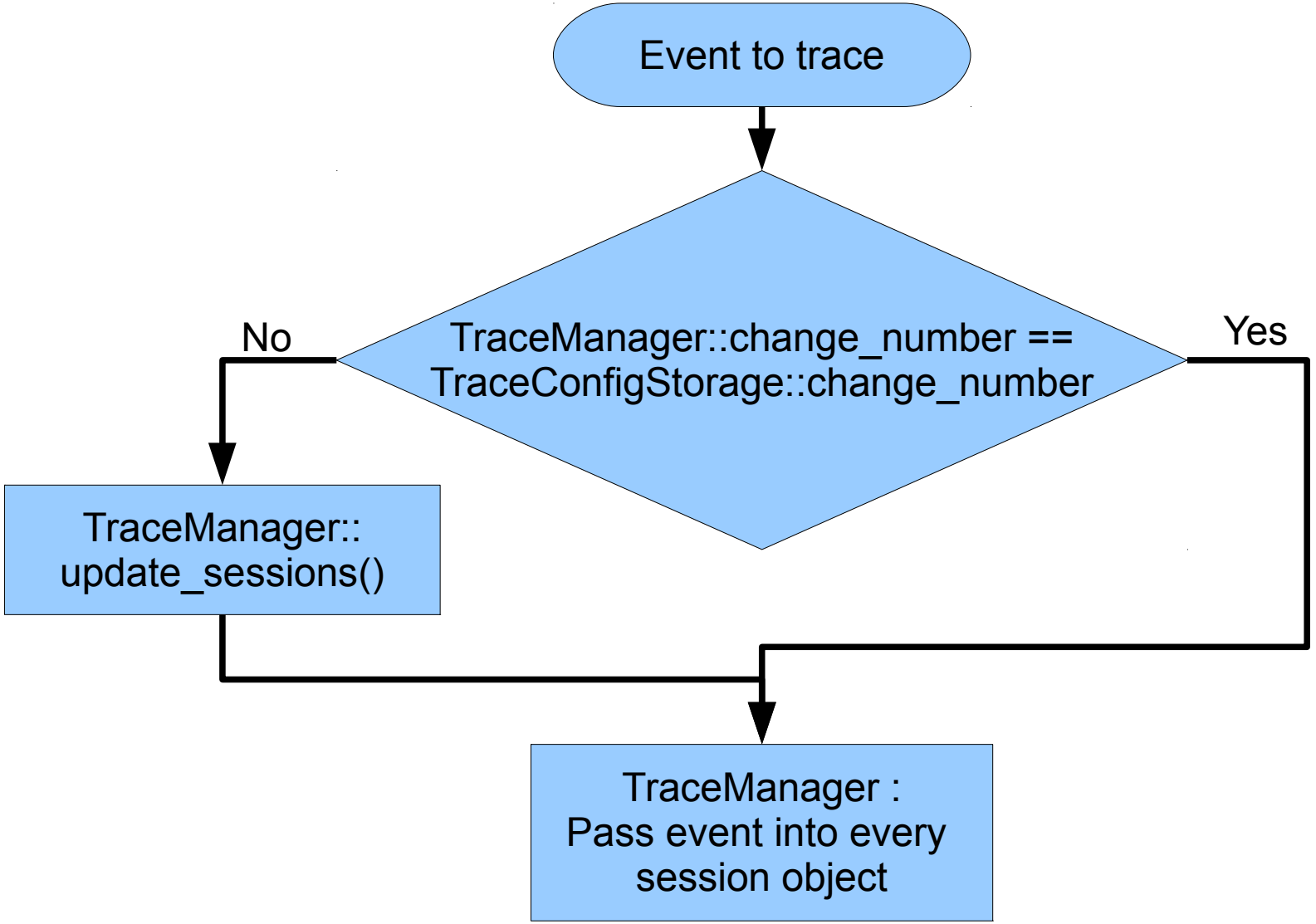
Trace Sessions in the Engine

- All trace activity in the engine done using TraceManager object
  - Each database connection has own instance of TraceManager
  - Each service connection has own instance of TraceManager
  - No additional synchronization between any kind of connections is required
- TraceManager sits between the engine and trace plugin(s)
  - Load and initialize trace plugin
  - Put events data into trace plugin methods

Trace Manager

- Maintains list of active trace session objects which should be used at database\service connection
- Maintains list of trace plugin's which should receive corresponding events
- Two main tasks
  - React on changes in trace sessions list
    - New trace session created
    - Requests to stop trace session
    - Requests to suspend\resume trace session
  - Pass events produced by the engine into trace plugin(s)
- Should have minimal impact on engine performance!

Trace Manager



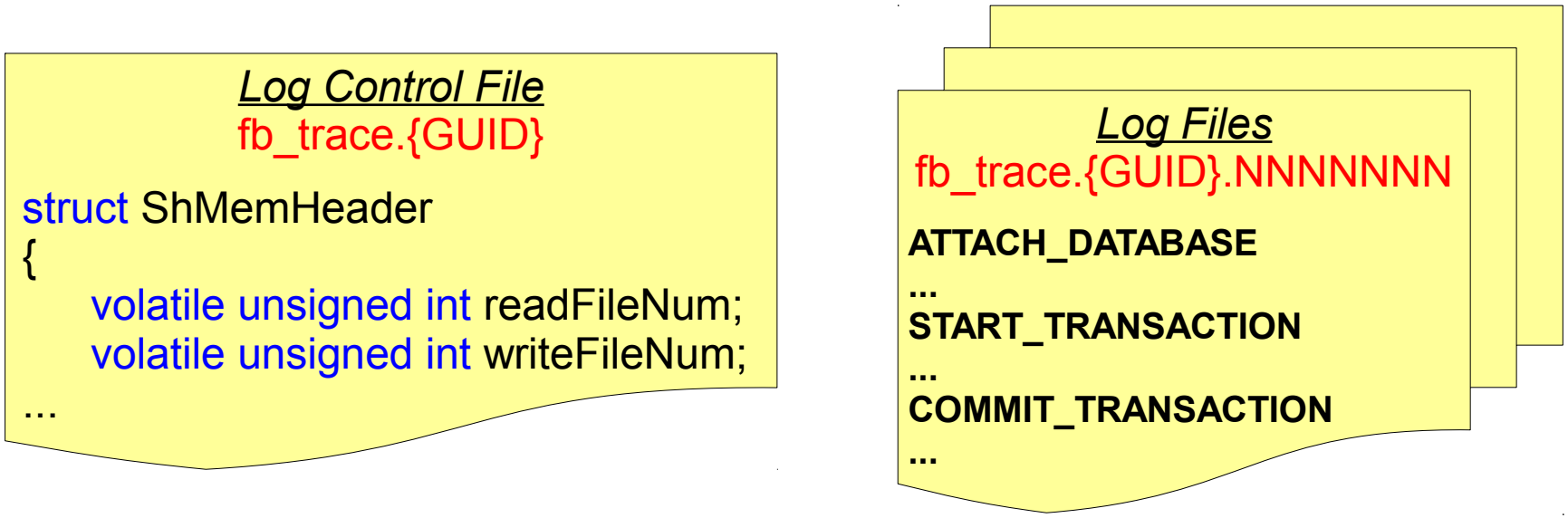
Trace session's output

- User trace
  - Many writers
    - Other connections, produced events to trace
  - One reader
    - The service connection which creates trace session
- It is hard to directly pass events data from different engine processes into given service connection
- Writers needs to store it somewhere and reader should read it
- Disk space used should be limited

Trace session's output

- User trace
  - Many writers
  - One reader
  - Disk space released while reading
  - Delete whole log when reader gone
  - Log file name set by Firebird
- System audit trace
  - Many writers
  - No readers
  - Log files rotation
  - Log files not deleted by Firebird
  - Log file name set in trace configuration on per-database (service) basis

Output of user trace



- Log files placed at Firebird's lock directory
  - by default COMMON_APPDATA\firebird
- Maximum size of each log file is 1MB
- Maximum summary log size set in firebird.conf
  - MaxUserTraceLogSize = 10

Output of System Audit trace

- Stored in disk file(s)
- File name is set in trace configuration file on per-database (per-service) basis
  - «log_filename» setting in *database* or *service* section
  - Each traced database or service could have own trace log file
- Log file could be “rotated” when its size reached «max_log_size» MB
  - Current log file is renamed
    - log_filename.<timestamp>
  - New log file is created
  - Old log files should be removed by the administrator

Trace tips and hints

- The more events is traced the more performance delay could happen
- Be careful:
  - Trigger related events will appear for every record affected by DML statements!
  - If you use some PSQL function in SQL statement – function execution could be traced as many times as records processed!
  - Same about joins with stored procedures
  - Same about RDB$SET_CONTEXT usage

Trace tips and hints

- Statements like
  EXECUTE PROCEDURE, or
  SELECT ... FROM<procedure>
  will produce
  - set of SQL statement related events
    - log_statement_prepare
    - log_statement_free
    - log_statement_start
    - log_statement_finish
  - set of PSQL related events
    - log_procedure_start
    - log_procedure_finish
- Choose what you really need

Trace tips and hints

- Question
  - Database could be accessed by file name or by alias, or
  - I need to trace same set of events for few databases using one trace session
- Solution
  - set all required events and params at default *database* section, but do not **enable** it
  - add few more *database* sections using patterns and set enable = true there

Trace tips and hints

```
database
{
    enabled = false
    log_statement_finish = true
    print_perf = true
    time_threshold = 60
}

# by the file name pattern
database = c:\\databases\\%.fdb
{
    enabled = true
}

# by database alias
database = mycooldb
{
    enabled = true
}
```


Trace tips and hints

- Consider to “mark” interesting queries using comments embedded into query text and later add filter at trace configuration to include (or exclude) such queries:

```
/* no trace me */ SELECT …
```

exclude_filter = %(no trace me)%

- Big app with a lot of modules\queries:
mark queries with module names to easy find source module which run the query

Trace tips and hints

- Performance statistics in trace and in tools (isql, etc) could be different
- Trace shows “clear” time spent by the engine only to execute statement
- ISQL time includes
  - network delays
  - prepare time
  - commit time (DDL statements and autoddl = on)

Trace tips and hints

- Performance statistics in trace and in tools (isql, etc) could be different
- How ISQL measure stats and time:
  - take "before" stats
  - free previous stmt
  - prepare current stmt
  - ask for PLAN (if required)
  - execute statement
  - ask for record counts (if query is ins\upd\del)
  - fetch and print results
  - take "after" stats
  - print difference "after" - "before"

Trace tips and hints

- Log of slow queries

```
database
{
  log_statement_finish = true
  print_perf = true
  time_threshold = 10000
}
```


Trace tips and hints

- Log DDL statements

```
database
{
  log_statement_start = true
  include_filter = %(CREATE|ALTER|DROP)%
  time_threshold = 0
}
```

case insensitive filter could looks a bit more complex

```
%([Cc]Rr[Ee][Aa][Tt][Ee])%
```


Trace tips and hints

- Trace configuration uses regexp patterns with the same syntax as Firebird itself
- You may check regexp pattern\filter with Firebird:

  SELECT ‘Matched’ FROM RDB$DATABASE
  WHERE <string> IS SIMILAR TO <pattern string>

THANK YOU FOR ATTENTION !

Questions ?

Firebird official web site

Firebird tracker

hvlad@users.sf.net