29 ноября

# ТЕХНОЛОГИИ БАЗ ДАННЫХ

## Третья практическая конференция

Организатор

ОТКРЫТЫЕ СИСТЕМЫ
Open Systems Publications

# Firebird:

## мониторинг, трассировка и диагностика

Дмитрий Еманов
dimitr@firebirdsql.org

Firebird Project
www.firebirdsql.org

# Что имеет смысл мониторить

- Доступность и целостность
- Кто и что делает
- Качественные и количественные данные
- Пиковые и предельные ситуации
- Системные и прикладные ошибки

# Доступность и целостность

- Доступность — извне, на разных уровнях
- Целостность — изнутри:
  - Критические ошибки в логах СУБД
  - Опционально: контрольные суммы
  - Возможность онлайн-валидации физической структуры

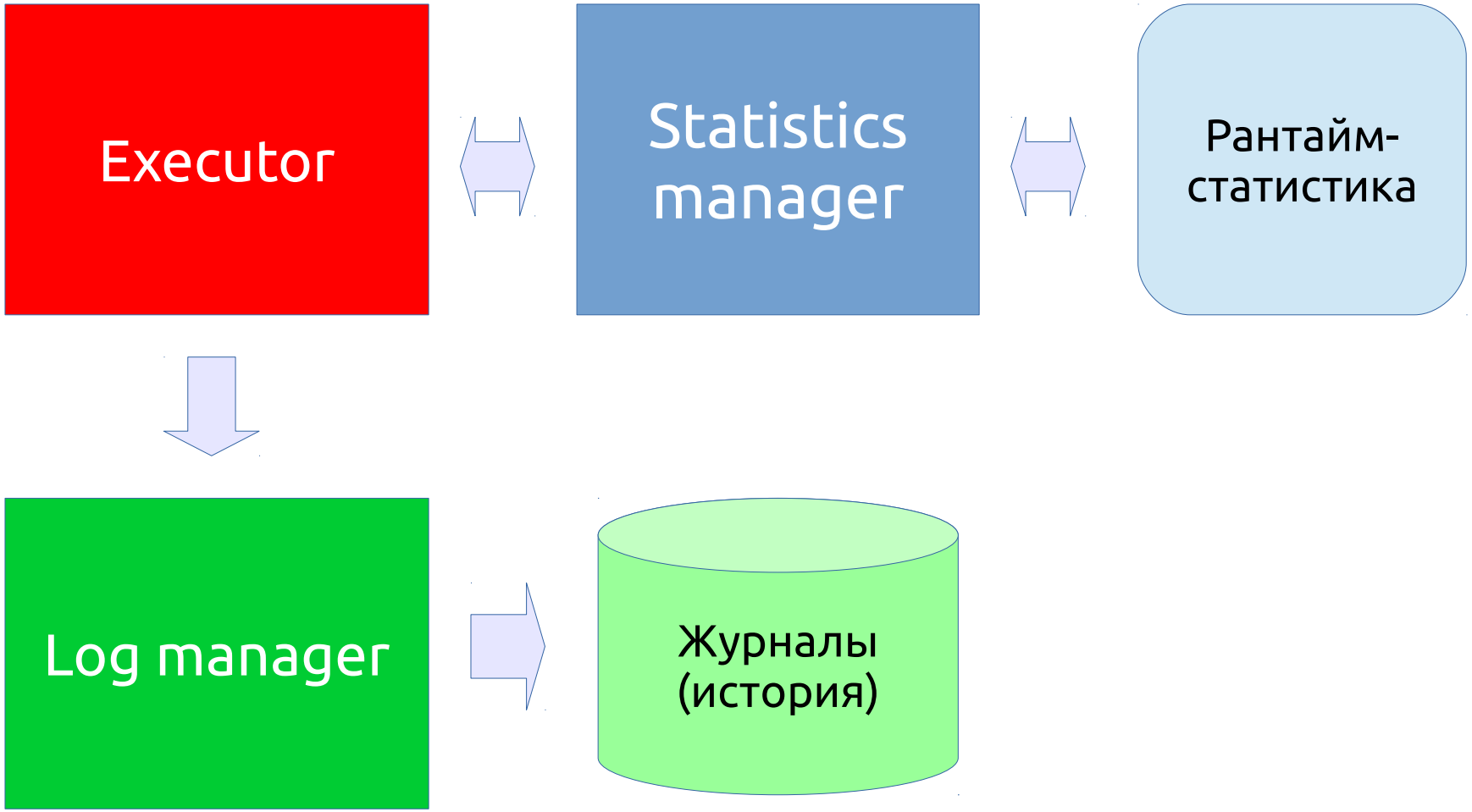
# Кто и что делает

- Активность (а также псевдо-активность) юзеров
- Привязка к коннектам, транзакциям, SQL-запросам
- Идентификация клиентов
- Привязка к процессам / потокам ОС
- Фоновые задачи СУБД

# Идентификация клиентов в Firebird

- С какого хоста пришел запрос:
  host/IP, транспортный протокол
- С какого приложения пришел запрос:
  port, PID, process pathname, tag
- Каким API пользовался клиент:
  версия клиентской либы, версия протокола

# Какая информация бывает



# Какая информация бывает

Рантайм-статистика

- Что происходит прямо сейчас (на момент обращения), текущие значения метрик, иногда пиковые значения метрик
- Требует периодического опроса; может пропускать события; высокочастотный опрос может сильно нагружать СУБД

# Какая информация бывает


- Полная или частичная хронология событий мониторинга
- При злоупотреблении нагружает СУБД; быстро забивает дисковое пространство; надо четко понимать, что именно хотим знать

# Инструменты в Firebird

- Онлайн-валидация — через API/консоль
- Метрики своего коннекта — через API
- Таблицы рантайм-мониторинга — через SQL
- Трассировка/аудит — через API/консоль
- Состояния блокировок — через консоль

# Метрики

- Обычно это текущие значения счетчиков
- С одной стороны, считать надо много событий
- Но их детализация может стоить недешево

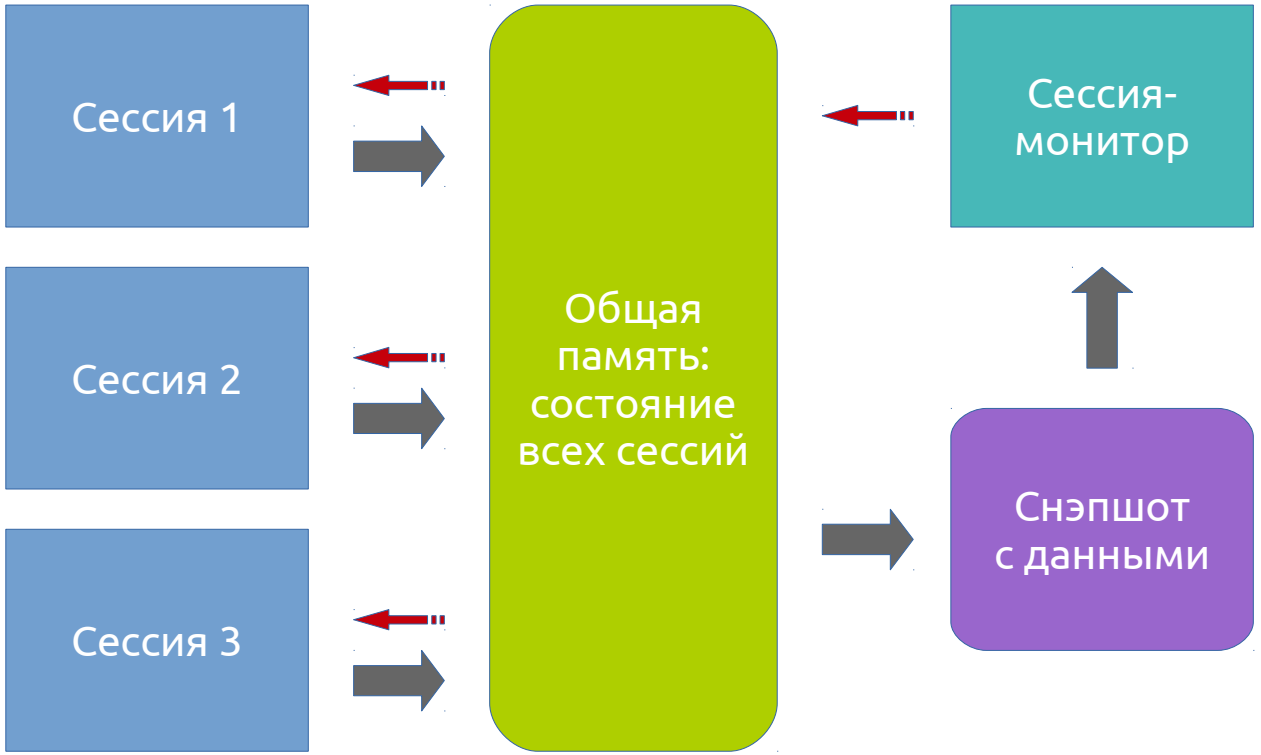
# Детализация метрик

- Только для сессии?
  Или еще и для каждой транзакции?
  Или еще и для каждого запроса?
  А может, еще ниже уровнем?
- Кое-что надо бы потаблично
  А еще поиндексно...

# Объекты мониторинга в Firebird

- База данных, сессия, транзакция, DSQL-запрос, вложенные PSQL-вызовы
- Каждый уникально идентифицирован
- Объекты связаны друг с другом
- Каждый объект владеет полным набором метрик

# Архитектура рантайм-мониторинга



# Архитектура рантайм-мониторинга

- Виртуальные таблицы мониторинга (данные хранятся в памяти)
- Фоновые задачи представлены отдельными системными сессиями
- Метрики двух видов — инкрементные счетчики и текущее/пиковое значение ресурса
- Метрики считаем всегда (локально)

# Архитектура рантайм-мониторинга

- Полная информация собирается по запросу (при обращении к MON$-таблицам)
- Снэпшот на время жизни транзакции
- Обновление снэпшота возможно через автономные транзакции

# SQL-интерфейс к мониторингу

- MON$DATABASE
- MON$ATTACHMENTS
- MON$TRANSACTIONS
- MON$STATEMENTS
- MON$CALL_STACK
  +
- MON$*_STATS
- MON$*_USAGE

# SQL-интерфейс к мониторингу

- MON$DATABASE
- MON$ATTACHMENTS
- MON$TRANSACTIONS
- MON$STATEMENTS
- MON$CALL_STACK

+

- MON$*_STATS
- MON$*_USAGE

Считаются для каждого объекта выше

# SQL-интерфейс к мониторингу

- MON$DATABASE
- MON$ATTACHMENTS
- MON$TRANSACTIONS
- MON$STATEMENTS
- MON$CALL_STACK

+

- MON$*_STATS
- MON$*_USAGE

Агрегируются
снизу вверх
в реальном времени

Считаются для
каждого объекта
выше

# Мониторинг ожиданий

- I/O, блокировки разных видов
- Кто, кого и на чем ждет
- Интересен также контекст
- Графы ожиданий, дэдлоки

# Мониторинг ожиданий в Firebird

- Есть мощная утилита fb_lock_print
- Вывод текстовый, под нагрузкой очень большие объемы, тяжело парсить
- Требует понимания внутренней кухни
- Интегрируем менеджер блокировок с мониторингом через MON$-таблицы

# Требования к логированию

- Критические ошибки
- Алерты — логические ошибки и пиковые ситуации (медленные запросы, временные файлы)
- Интерактивная трассировка
- Аудит — постоянное логирование

- Хотелось иметь единое гибкое решение

# Трассировка в Firebird

- Trace API + плагины
- Системный аудит — управляется ядром, настраивается конфигом, пишется в лог на диске (по умолчанию) с ротацией (архивированием)
- Пользовательская трассировка — управляется юзером (start-pause-resume-stop), конфиг задается в рантайме, лог пишется во временные файлы и удаляется по мере вычитывания клиентом

# Трассировка в Firebird

- Системный аудит используется для аудита безопасности, а также для непрерывного мониторинга производительности (с высоким порогом)
- Пользовательская трассировка используется для профилирования нагрузки в продакшене, а также для более детального мониторинга (с меньшим порогом или без него)

# Концепция трассировки

- Везде храним ID объектов
- Большинство событий спаренные (начало / конец)
- Для SQL-запросов можем показывать планы
- По завершении можем выдавать статистику — время выполнения и детализированные метрики

# Концепция трассировки

- Можно трассировать прикладные ошибки
- Можно трассировать вызовы процедур/функций/триггеров
- Есть функционал slow query log — через time_threshold
- Дополнительная кастомизация — через include/exclude фильтры (regexp)

# Трассировка: примеры

```
# трассируем медленные запросы
database = myslowdb
{
    log_statement_finish = true
    print_perf = true
    time_threshold = 10000      # 10 seconds
}


# трассируем изменения метаданных
database = mycooldb
{
    log_statement_start = true
    include_filter = %(CREATE|ALTER|DROP)%
}
```

```
2017-10-12T08:01:39.2930 (26308:0x7f0b97c6c978) EXECUTE_STATEMENT_FINISH
arkandus (ATT_23, ALEX12:NONE, UTF8, TCPv4:127.0.0.1)
(TRA_283505, READ_COMMITTED | REC_VERSION | WAIT | READ_WRITE)

Statement 323352:
-------------------------------------------------------------------------------
select * from SP$GET_STANDARD_BILL_REPORT(?, ?, ?)

PLAN (ANREDE INDEX (PK$ANREDE))(ANREDE INDEX (PK$ANREDE))(BILL_CREDITS INDEX …
PLAN (K BG INDEX (RDB$PRIMARY245)), M_U1 INDEX (PK$PERSON)), A1 INDEX (PK$ANREDE)) …
PLAN JOIN (JOIN (UK U INDEX (UNQ_USERS), UK A INDEX (PK$ARZT)), UK PA INDEX …

param0 = integer, "174925"
param1 = integer, "<NULL>"
param2 = integer, "<NULL>"

1 records fetched

   1068 ms, 14135 read(s), 664071 fetch(es)
```

| Table | Natural | Index | Update | Insert | Delete | ... |
|---|---|---|---|---|---|---|
| ACCOUNT_CATALOGS | | 1 | | | | |
| ANREDE | | 2 | | | | |
| ARZT | | 1 | | | | |
| BILL_PARTS | | 13 | | | | |
| BILLING | | 1 | | | | |
| CL_BILL_LENS_CATEGORIES | | 2 | | | | |
| CL_BILL_LENSES | | 2 | | | | |
| COST_ESTIMATES | | 1 | | | | |
| DIVISIONS | | 2 | | | | |
| KV_DMP | | 1 | | | | |
| MITARBEITER | | 1 | | | | |
| OPTIONS | | 3 | | | | |
| PATIENT_INSURANCES | 317831 | 4 | | | | |
| PATIENTEN | | 1 | | | | |
| PERSON | | 2 | | | | |
| USERS | | 2 | | | | |
| HOLIDAY_DIVISIONS | | 120 | | | | |
| HOLIDAYS | | 120 | | | | |

# Интегрированный подход

- В трейсе — алерты + базовые события + пороговые события
(все это пишем в append-only database)
- Среднечастотный опрос таблиц мониторинга
- При разборе полетов связываем данные вместе по их ID —> по текущему состоянию вытягиваем предысторию

# Вопросы?