X5 RETAIL GROUP

Российский рынок продуктового ритейла

Рынок привлекательный, несмотря на растущую конкуренцию

№8

в мире по объему
191 млрд $

73%

доля современной розницы

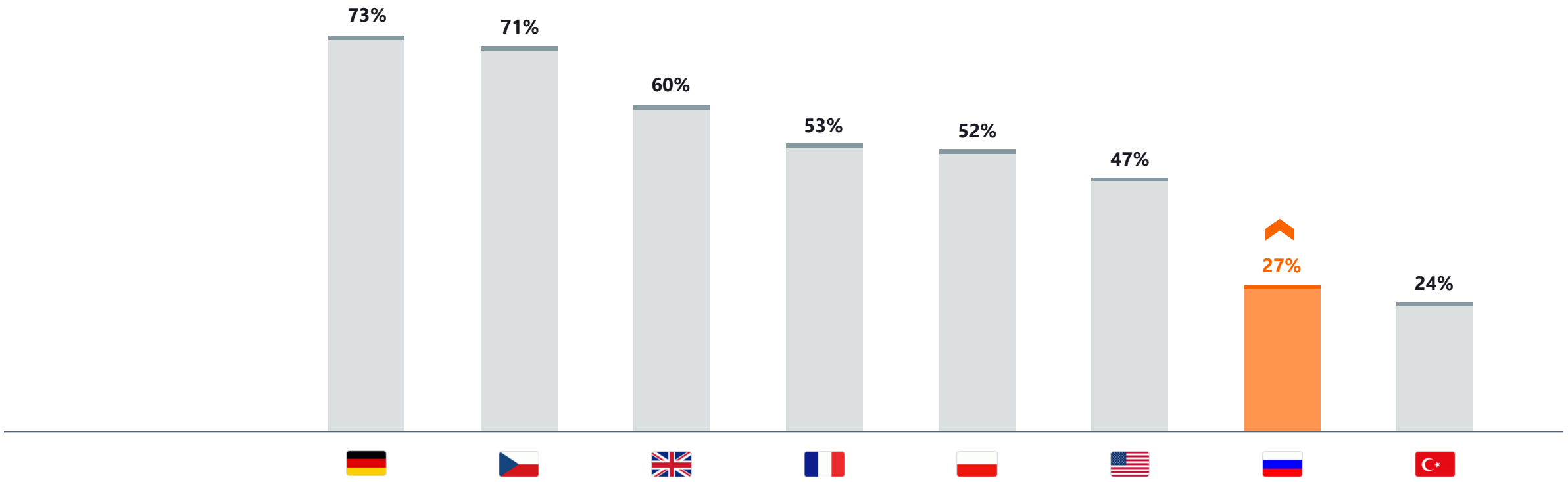
27%

доля топ-5 игроков

Большой потенциал для крупнейших игроков

Консолидация рынка — основной драйвер роста

Доля топ-5 игроков продуктового рынка в 2019 году



Источники: Euromonitor, Infoline (для России)

Тенденции российского рынка продуктовой розницы

X5 RETAIL GROUP

Business Monitoring System (BMS) как средство анализа взаимодействия с покупателем

Как сделать business-process mining на неструктурированных POS логах средствами R

X5 RETAIL GROUP

Ролдугин Евгений

Начальник управления

Управление развития и сопровождения решений автоматизации, мониторинга,аналитики и новых технологий

X5 Retail Group

Ключевые различия классического ИТ мониторинга и бизнес-мониторинга

Классический мониторинг

- Сбор отдельных самодостаточных метрик
- Сильная техническая направленность метрик
- Минимальная аналитическая обработка собранных данных

Бизнес-мониторинг

- Сбор событий
- Сборка из событий транзакций
- Вычисление производных показателей, присущих только транзакциям
- Транзакционное окно средней продолжительности (от секунд до десятков минут)

Общие характеристики проекта

- Срок исполнения ~ 1,5 года
- Ядро команды ~ 5 чел.

- ~ 15 К магазинов на старте
- ~ 3,5 касс\магазин (в среднем)
- ~ 150 Мб raw логов с кассы в сутки
- ~ 1 год хранения сырых данных
- 5 часовых зон

"*Зачастую совершенствование того или иного ИТ-процесса никак не заметно обычному покупателю. Наше изобретение помогает клиентам экономить время: ускоряет операции на кассе, сокращает очереди и остановки на кассе. Речь идёт о секундных задержках, но на 4 млрд чеков, которые за год печатаются в сетях X5, 1 секунда превращается в 98 лет экономии. Результаты работы всей команды позволяют нам говорить о том, что мы ускоряем время обслуживания на кассе, оптимизируем технические операции и влияем на такие важные показатели компании как NPS и товарооборот*", - говорит ВРИО директора по информационным технологиям X5 Retail Group Алла Антонова.

http://www.comnews.ru/digital-economy/content/203748/2019-12-24/2019-w52/razrabotka-it-x5-uskorit-kassovye-operacii

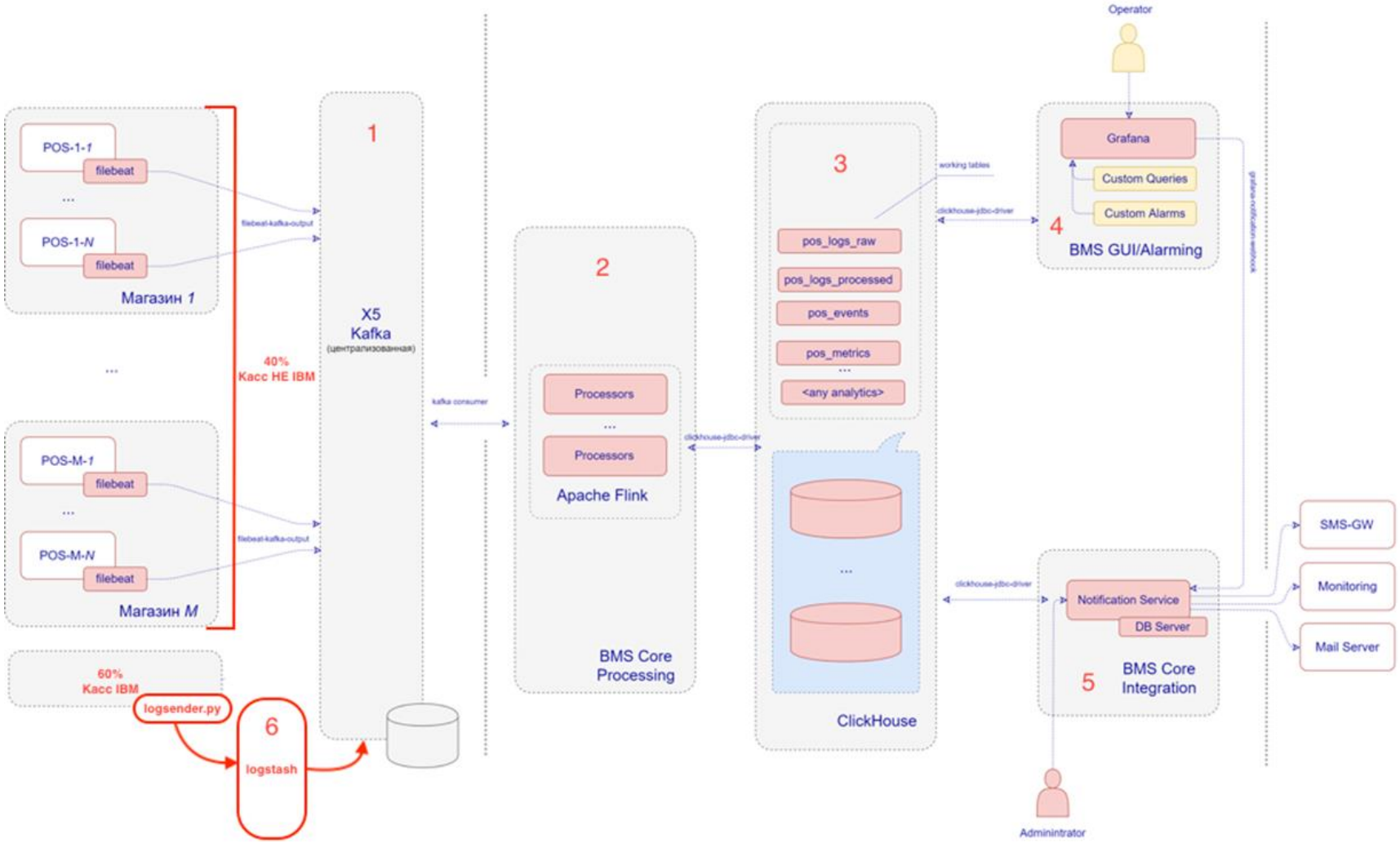
Как выглядит POS лог

```
edit std_pos.log - Far 3.0.4774 x64
D:\X5_LOGS\GK logs\gk-logs-2019-10-30\POS01-BO-335N\std_pos.log                                                    1251  Ln  33124/33278 Col 1    Ch 1        50          12:12
2019-10-31 10:16:43:695 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] directIO(command..81, data..null, object..[])
2019-10-31 10:16:43:695 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl] CALL - directIO(command..81, data..null, object..[])
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl] RETN - directIO() : TRUE
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[0]: 3
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[1]: 1
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[2]: 31.10.2019T09:58
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[3]: 76849
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[4]: 1352411847
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[5]: 1
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl_SHTRIH_MFRK] createFiscalReceipt()data[6]: 24754
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl] CALL - getPropertyValue(code..3020)
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl] RETN - getPropertyValue() : 2
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [PlGkBonFiscal] [storedFiscalDocNr = null; lastFiscalDocNr = 76849] => printedOK = false
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl] CALL - getFiscalPrinterState(state .. [0])
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [FiscalPrinterImpl] RETN - getFiscalPrinterState() : TRUE [1]
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [AbstractPrintoutManager] Initialization of section report started
2019-10-31 10:16:43:746 DEBUG <main> {AWT-EventQueue-0} [PosTemplateProvider] Loaded template JOO2_bon.xml within 0 ms.
2019-10-31 10:16:43:782 DEBUG <main> {AWT-EventQueue-0} [TbiRowExtDataHandler] Row type gk.std_pos_repl.data.db.stamm.tbi.TbiEmployeeRow is not extensible
2019-10-31 10:16:43:782 DEBUG <main> {AWT-EventQueue-0} [TbiRowExtDataHandler] Row type gk.std_pos_repl.data.db.stamm.tbi.TbiEmployeeRow is not extensible
2019-10-31 10:16:43:782 DEBUG <main> {AWT-EventQueue-0} [ReportGeneratorImpl] Output device is 'JournalOutputDevice' with resolution 1 DPI and page size gk.rg.util.Dimension[width=57,height=2048] pixels
2019-10-31 10:16:43:783 DEBUG <main> {AWT-EventQueue-0} [AbstractPrintoutManager] Initialization of section report finished
2019-10-31 10:16:43:783 DEBUG <main> {AWT-EventQueue-0} [JournalOutputDevice] Journal NOT saved to beleg. Nothing has been printend to journal.
2019-10-31 10:16:43:783 DEBUG <main> {AWT-EventQueue-0} [ReportManagerImpl] Section report BON finished
2019-10-31 10:16:43:783 DEBUG <main> {AWT-EventQueue-0} [DStoredDatabaseManager] jdbc:firebirdsql:localhost/3052:/usr/local/gkretail/pos/data/standard_beleg.gdb?encoding=UTF8
2019-10-31 10:16:43:783 DEBUG <main> {AWT-EventQueue-0} [ToReceiver] ToReceiver.resume called
2019-10-31 10:16:43:783 DEBUG <transportobjectconsumer> {ClientConsumer} [ToReceiver] transportobjectconsumer status: Working
2019-10-31 10:16:43:885 INFO  <main> {APRScanKeysThread} [AprKeyboardServiceImpl] APR-KB Listeners kb is ON - Point02 -> code is #44 at code is #61
2019-10-31 10:16:43:885 INFO  <main> {APRScanKeysThread} [SZLifKeyboard$ScanKeys] sc_key:   #[ 00 00 2C 00 00...] str:
2019-10-31 10:16:43:886 DEBUG <main> {Thread-1143} [JPosKeyboardHP] HP-KB: HPKeyboardJposAdapter processData p1. hexData = 3D, null
2019-10-31 10:16:43:887 INFO  <main> {APRScanKeysThread} [AprKeyboardServiceImpl] APR-KB Listeners kb is ON - Point02 -> code is #0 at code is #0
2019-10-31 10:16:43:887 INFO  <main> {APRScanKeysThread} [SZLifKeyboard$ScanKeys] sc_key:   #[ 00 00 00 00 00 00...] str:
2019-10-31 10:16:43:901 INFO  <main> {APRScanKeysThread} [AprKeyboardServiceImpl] APR-KB Listeners kb is ON - Point02 -> code is #98 at code is #99
2019-10-31 10:16:43:901 INFO  <main> {APRScanKeysThread} [SZLifKeyboard$ScanKeys] sc_key:   #[ 00 00 62 00 00...] str: 0
2019-10-31 10:16:43:901 DEBUG <main> {Thread-1144} [JPosKeyboardHP] HP-KB: HPKeyboardJposAdapter processData p1. hexData = 63, 10
2019-10-31 10:16:43:901 DEBUG <main> {Thread-1144} [JPosKeyboardHP] HP-KB: HPKeyboardJposAdapter processData p2. dataInt = 10
2019-10-31 10:16:43:901 DEBUG <main> {Thread-1144} [JPosKeyboardHP] HP-KB: HPKeyboardJposAdapter before super.processData. dataInt = 10
2019-10-31 10:16:43:903 INFO  <main> {APRScanKeysThread} [AprKeyboardServiceImpl] APR-KB Listeners kb is ON - Point02 -> code is #0 at code is #0
2019-10-31 10:16:43:903 INFO  <main> {APRScanKeysThread} [SZLifKeyboard$ScanKeys] sc_key:   #[ 00 00 00 00 00 00...] str:
2019-10-31 10:16:43:914 DEBUG <transportobjectconsumer> {ClientConsumer} [PlToConsumer] Saving transaction transport object: [1000000000000954602, 12.08.49-SP33, 2019-10-31 09:05:51.486] ToContainer:   3/16 [ToItemSalesRestriction(4): ([X]) D, ToI
2019-10-31 10:16:43:914 DEBUG <main> {AWT-EventQueue-0} [CustomerTenderChecker] Check of customer related tenders disabled - no refresh
2019-10-31 10:16:43:914 DEBUG <main> {AWT-EventQueue-0} [PlPrepinSession] validatePrepinSession - bon closed -> notification not necessary
2019-10-31 10:16:43:914 DEBUG <main> {AWT-EventQueue-0} [PlGkBonSignOffSession] Sign off after receipt processed is possible ( receipt type = 1 )
2019-10-31 10:16:43:914 DEBUG <main> {AWT-EventQueue-0} [PlAllowedTenderSession] Clearing temporary blocked tenders
2019-10-31 10:16:43:914 DEBUG <main> {AWT-EventQueue-0} [PlRefreshableUpdater] Update refreshable - STANDARD_MASTER_DATA_TYPE
2019-10-31 10:16:43:914 DEBUG <main> {AWT-EventQueue-0} [PlRefreshableUpdater] Refreshable name STANDARD_MASTER_DATA_TYPE is called in the master transaction.
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [PlRefreshableUpdater] Reloading component : name=PLI_TILL_ACCOUNTING, type=STANDARD_MASTER_DATA_TYPE
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [PlPositionClose] PositionClose : cleared opened bon
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [ExtensionPointCommandController] UcExtensionPoint [id=EP__DYNAMIC, adviceSet=[]]
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [UsecaseDynamicExtensionPointHandler] UcAdvice set: [UcAdvice [id=xrg_add_article_to_receipt]]
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [PosFunctionManagerImpl] PFM  popPosFunction - ru.x5.pos.core.function.item_registration.PosX_ItemRegistration_xrg [X_ITEM_REGISTRATION_5 (10094)][status=ERROR][output parameters -> empty]
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [PROCESS_POS_FUNCTION] X_REGISTRATION -> X_MAIN
2019-10-31 10:16:43:915 DEBUG <main> {AWT-EventQueue-0} [ViewManagerImpl] ViewManager: restore the view BonpreviewModel(13042929)<-BonpreviewScreenView
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [EventMgrImpl] Registering event observer ru.x5.pos.core.user_interface.mvc.specific.item_details.view.screen.bean.impl800.ItemDetailsAdditionalScreenPosBeanImpl$ItemInfoDetailPositionSel
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [ScannerProcessorImpl] resetInProcessMode()
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [AbstractResumeResolver] Start a resume resolver of PosX_Registration_xrg
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolveSignOffCommandController
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolveTillAccountingCommandController
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolvePaymentCommandController
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolveInternetOrderCollectionCommandController_xrg
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolveManualFiscalbelegCommandController
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.mvc.controller.ResolveGoodsReturnCommandController_xrg
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to gk.std_xpos.mvc.specific.result_message.ResolveResultMessageCommandController
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolveLocalAccountingCommandController
2019-10-31 10:16:43:916 DEBUG <main> {AWT-EventQueue-0} [UseCaseManagerImpl] Usecase PosX_Registration_xrg: redirect to ru.x5.pos.core.function.registration.resolver.ResolveTillPickupWarningCommandController_xrg
1Help   2Save   3    4Auto   5    6Hex    7Search   8Disk   9    10Quit   11Plugin   12Screen
```

Плюсы: POS – конечная точка взаимодействия с покупателем. Можно увидеть все узкие места и проблемные сценарии.

Минусы: Нет жесткой структуры, нет модели данных, нет сквозных идентификаторов, много лишнего шума, фрагменты транзакционных данных «размазаны» по множеству записей, лог служебный => допустима потеря фрагментов.

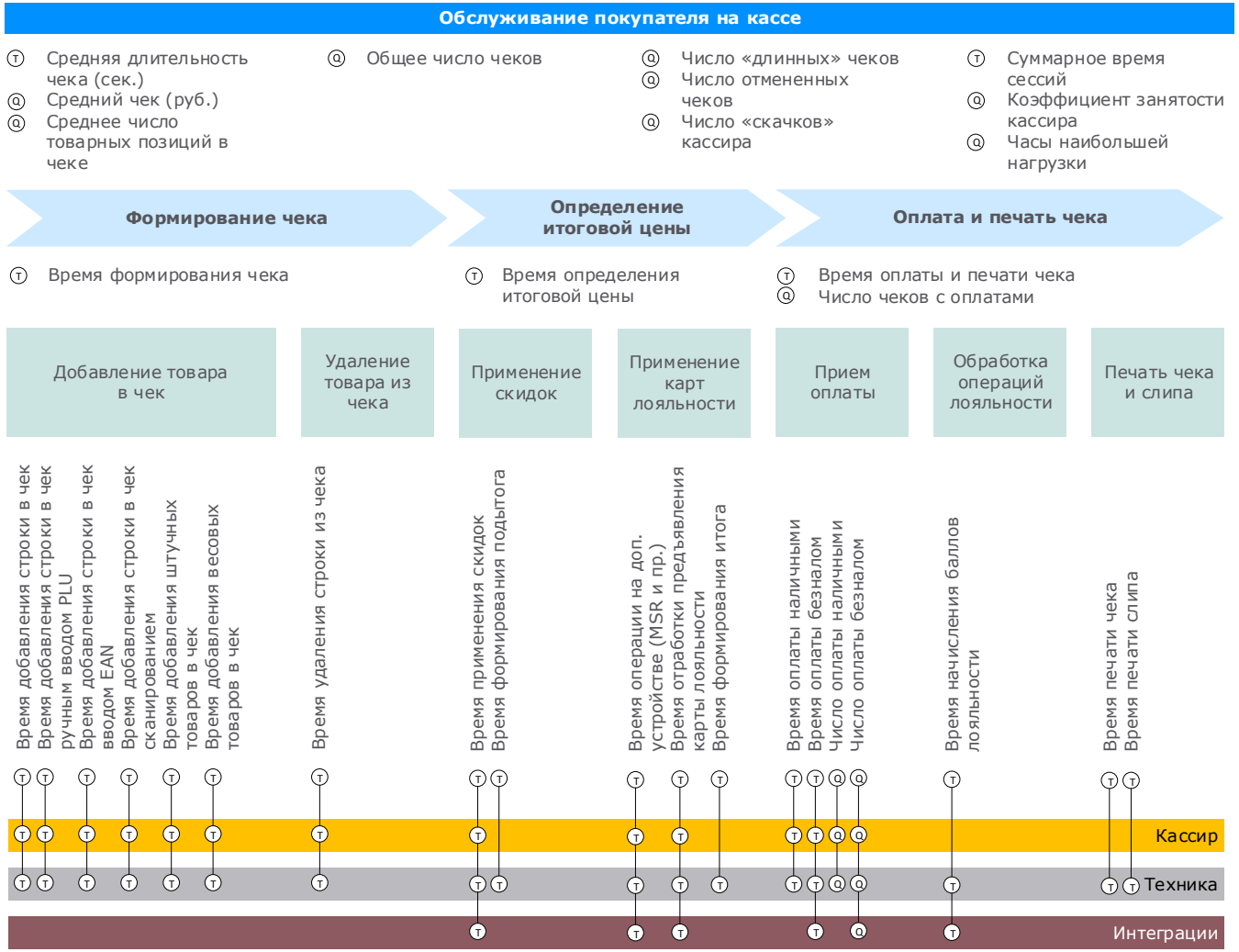
Архитектура BMS 1.0



Подсистемы BMS

- **X5 Kafka** – брокер сбора сообщений логов касс, реализован на Apache Kafka
- **BMS Core Processing** – обеспечивает потоковый сбор, обработку и передачу данных из X5 Kafka в БД ClickHouse. Реализован на Apache Flink. Используется нативный модуль интеграции Flink-Clickhouse-Sink
- **ClickHouse** – основное хранилище логов касс
- **BMS Core Integration** – сервис оповещений и интеграции БМС с внешними системами
- **BMS Analytics** – аналитическая подсистема, реализована на R
- **BMS GUI** – обеспечивает расчет КПЭ, установку порогов для оповещений и визуализацию дашбордов. Grafana\Shiny

Бизнес-мониторинг: контролируем параметры бизнес-процессов и бизнес-операций на POS



Process Mining.

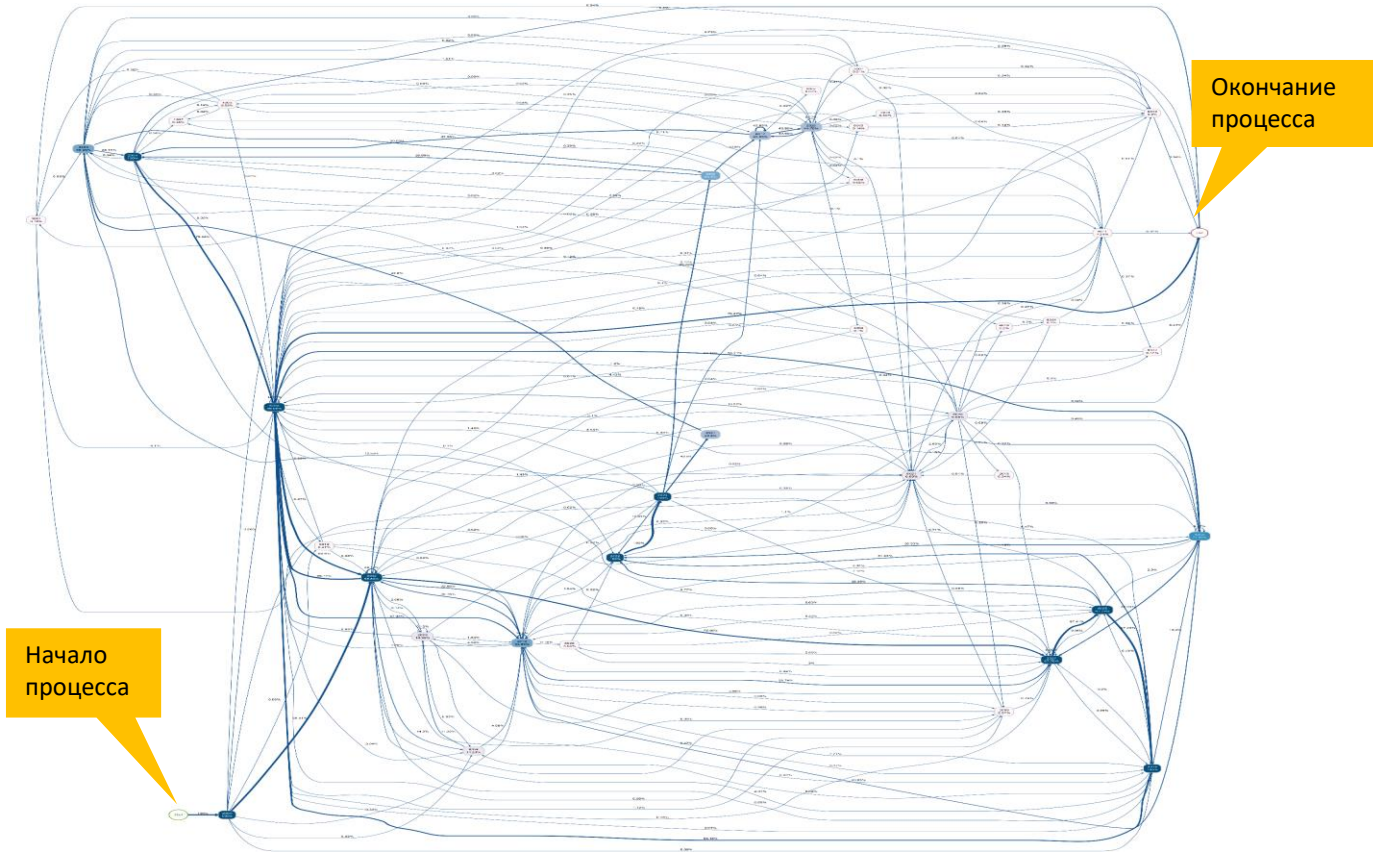
Пример карты переходов процесса кассового обслуживания

Источником информации являются журналы событий и таблицы логов, включающие дату и время события; описание события; идентификатор экземпляра процесса и другие аналитики процесса

С помощью подготовки данных, бизнес аналитики процесса и применения алгоритмов Process Mining и выполняется реконструкция реального процесса

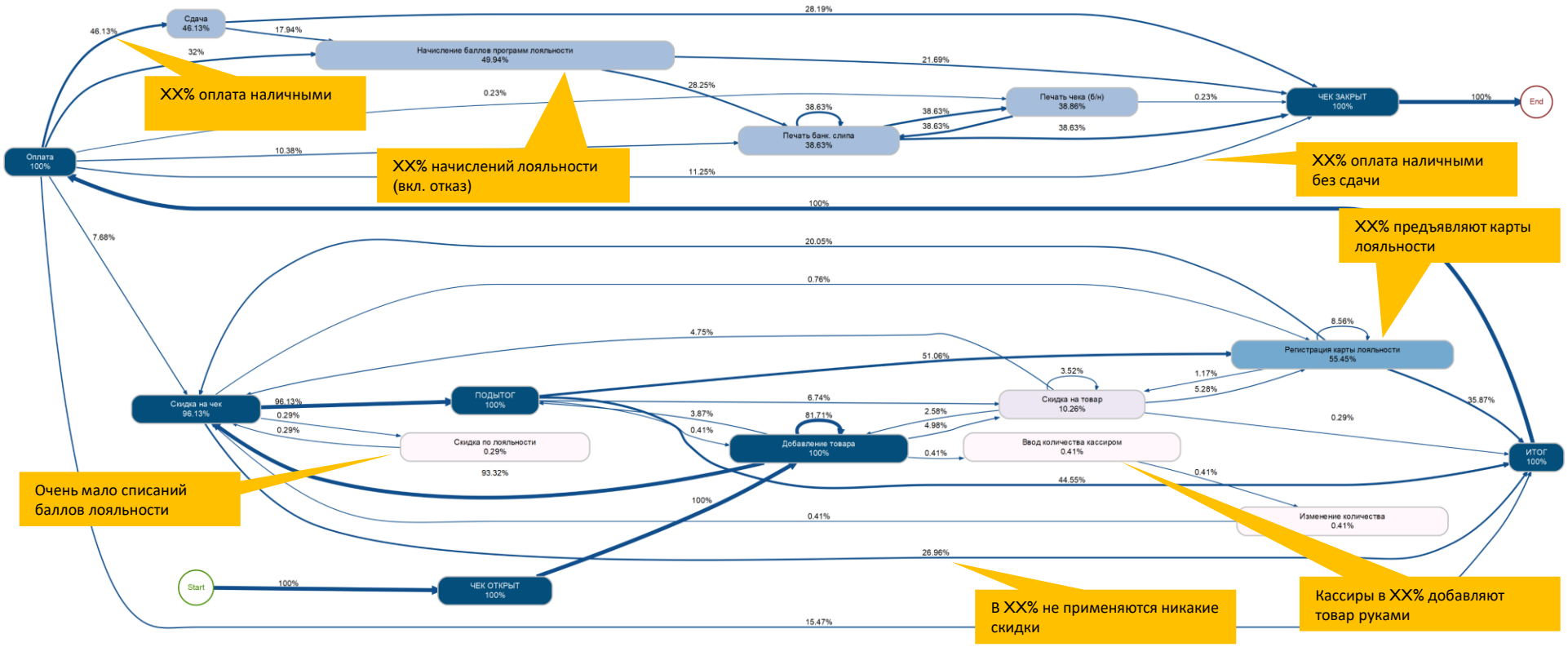
На базе построенной модели выполняются различные виды анализа: поиск «узких мест», анализ загрузки ресурсов, выявление внутренних циклов, переполнения очередей обработки, времен переходов, наиболее частых сценариев и др.

Аналитика данных CREP формата



Бизнес-отчетность по ИТ-сервисам

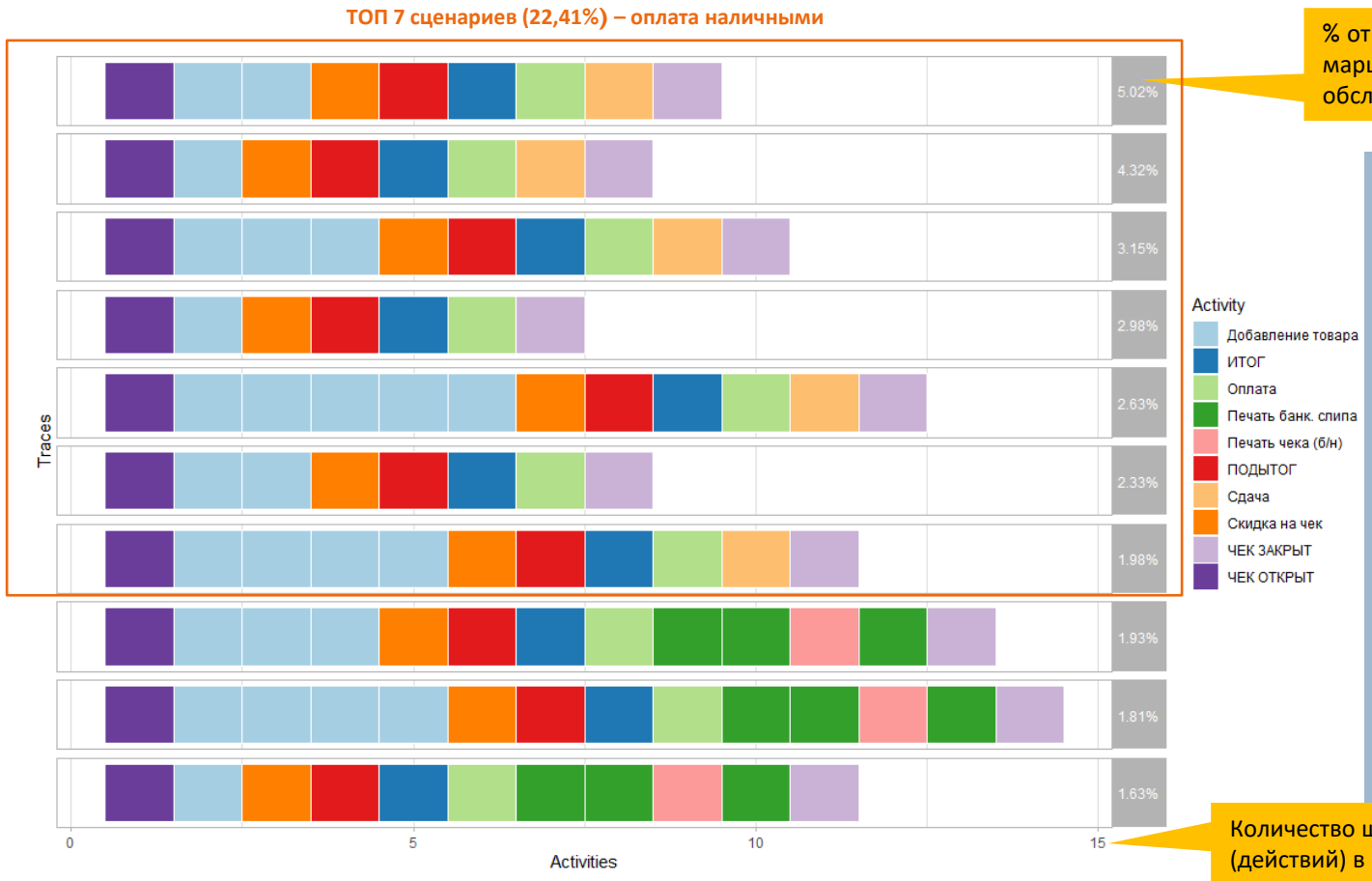
Вариант представления: карта частоты переходов между состояниями



Для удобства интерпретации карты переходов данные транзакций обогащены справочников кодов, выбраны ключевые для процесса обслуживания на кассе статусы и отключена визуализация несущественных (сточки зрения частоты событий) переходов.

Бизнес-отчетность по ИТ-сервисам

Процесс кассового обслуживания: ТОП 10 маршрутов



% от общего количества маршрутов процесса кассового обслуживания

КОММЕНТАРИЙ

ТОП 10 маршрутов суммарно закрывают 27,78% от общего числа возможных вариантов сессий кассового обслуживания.

Маршрут 1 (5,02%, 9 шагов)

1. ЧЕК ОТКРЫТ
2. Добавление товара
3. Добавление товара
4. Скидка на чек
5. ПОДЫТОГ
6. ИТОГ
7. Оплата (наличными)
8. Сдача
9. ЧЕК ЗАКРЫТ

Количество шагов (действий) в маршруте

X5 RETAIL GROUP

Герасева Алена

Начальник отдела развития и сопровождения инструментов мониторинга

Process mining как обязательный этап

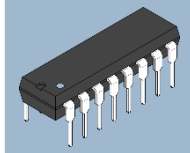
- Для расчета KPI необходимо от «зашумленных» событий перейти на уровень операций и транзакций.
- Классические Process Mining вендоры работают только со структурированными входными данными малого объема.
- Количество транзакций достигает $10^9 \cdots 10^{11}$ в сутки, приходится делать все самим.

CASE_ID	TIMESTAMP	EVENT
190311_1001_1_1	2019-03-11 08:00:06	INIT
193311_1001_1_1	2019-03-11 08:00:07	REQUEST
193311_1001_1_1	2019-03-11 08:00:10	RESPONSE
193311_1001_1_1	2019-03-11 08:00:14	ERROR
193311_1001_1_2	2019-03-11 08:00:48	INIT
193311_1001_1_2	2019-03-11 08:00:48	REQUEST
193311_1001_1_2	2019-03-11 08:00:49	RESPONSE
193311_1001_1_2	2019-03-11 08:00:49	ERROR
193311_1001_1_3	2019-03-11 08:01:28	INIT
193311_1001_1_3	2019-03-11 08:01:30	REQUEST
193311_1001_1_3	2019-03-11 08:01:33	RESPONSE
193311_1001_1_3	2019-03-11 08:01:37	ERROR
193311_1001_1_4	2019-03-11 08:01:58	INIT
193311_1001_1_4	2019-03-11 08:01:59	REQUEST
193311_1001_1_4	2019-03-11 08:02:02	RESPONSE
193311_1001_1_4	2019-03-11 08:02:06	ERROR
193311_1001_1_5	2019-03-11 08:02:15	INIT
193311_1001_1_5	2019-03-11 08:02:18	REQUEST
193311_1001_1_5	2019-03-11 08:02:20	RESPONSE
193311_1001_1_5	2019-03-11 08:02:24	ERROR
193311_1001_1_6	2019-03-11 08:02:26	INIT
193311_1001_1_6	2019-03-11 08:02:27	REQUEST
193311_1001_1_6	2019-03-11 08:02:28	RESPONSE
193311_1001_1_6	2019-03-11 08:02:30	ERROR

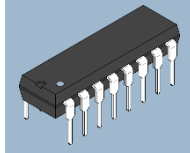
Аналитическое ядро

ClickHouse + R

На старте проекта

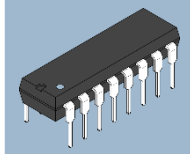


2 машины (шардирование)
RAM: 128 Gb
CPU: 16 core

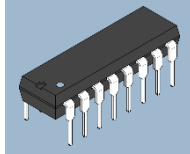


1 машина
RAM: 32 Gb
CPU: 4 core

В настоящий момент



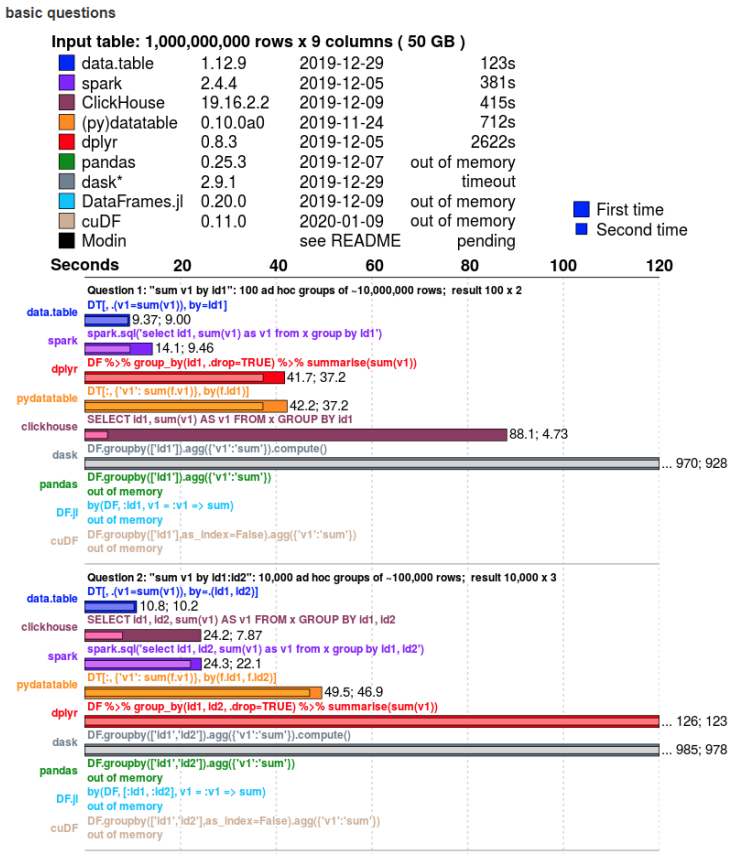
4 машины (шардирование)
RAM: 128 Gb
CPU: 16 core

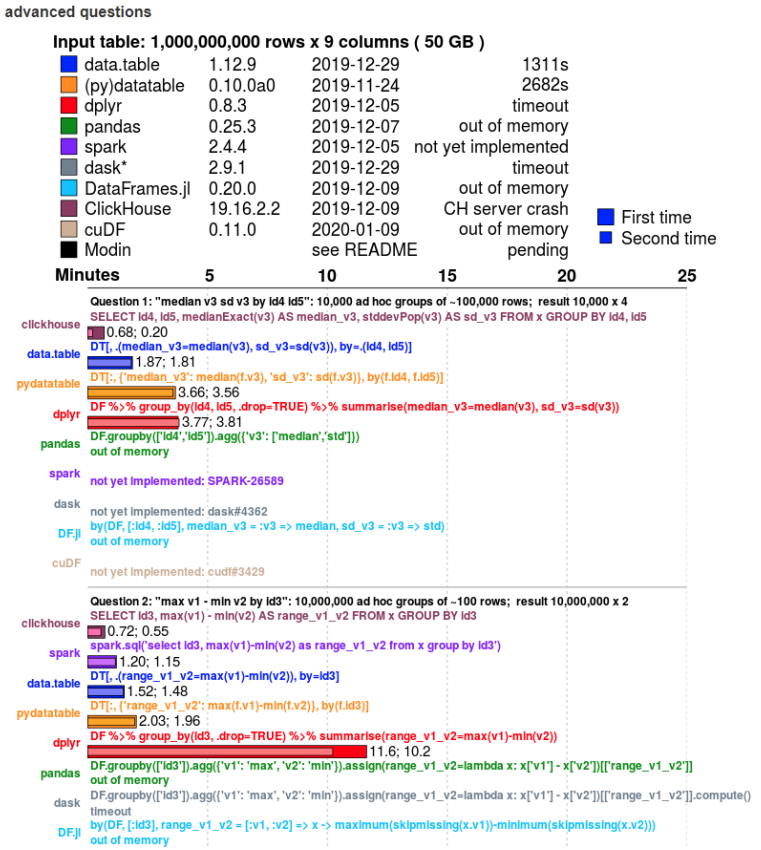


3 машины
RAM: 128 Gb
CPU: 16 core

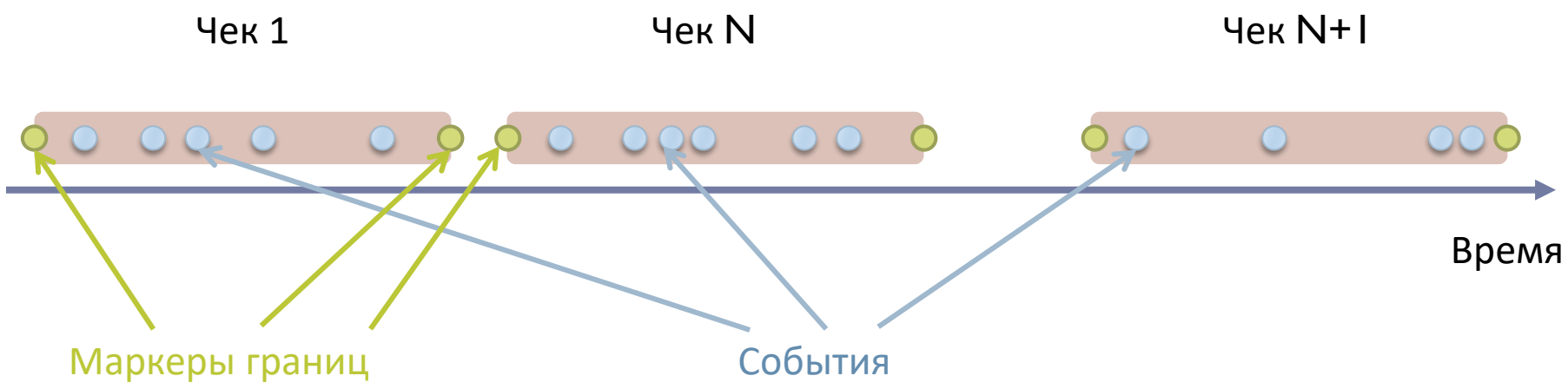
Data.table & Clickhouse benchmarks

Выбранный стек технологии явно хорош по производительности для задачи. Детали в «Database-like ops benchmark» (https://h2oai.github.io/db-benchmark/)





Алгоритм реконструкции транзакций



Идея: для каждой кассы будем двигаться по шкале времени и последовательно собирать все события между началом и концом чека в транзакцию

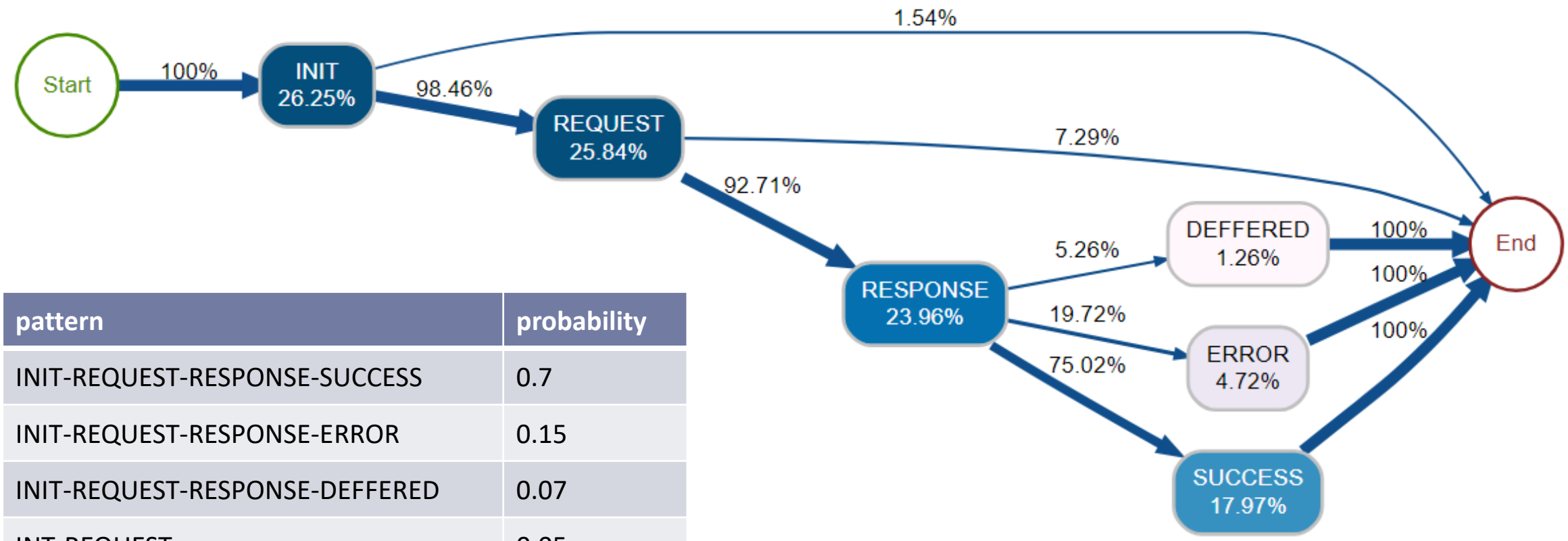
Но как это сделать быстро на наших масштабах?

Боремся за скорость

На объемах ~1 млрд событий и ~10 млн транзакций бездумными способами результат не получишь.

- О чем необходимо вспомнить:
  - Специфика работы с базовыми типами (особенно с POSIXct)
  - Работа с указателями, специфика выделения памяти
  - Сложность алгоритмов
  - Актуальные оптимальные реализации используемых функций
- Что категорически следует избегать?
  - Проводить вычисления по группам

Демо пример на основе системы лояльности



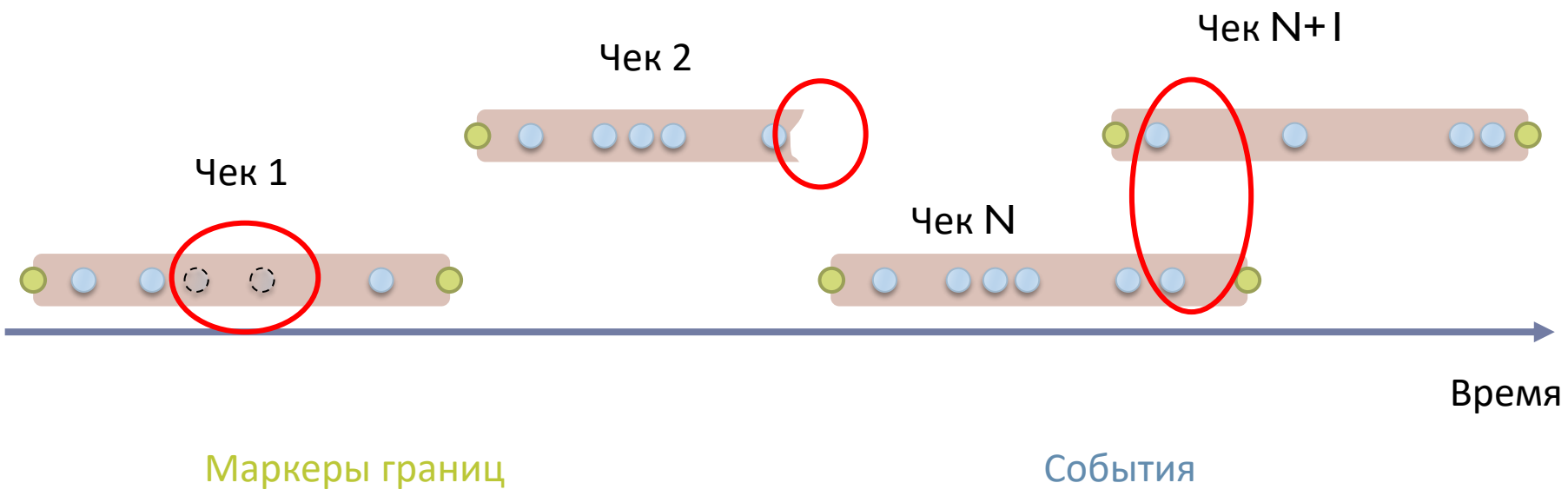
pattern	probability
INIT-REQUEST-RESPONSE-SUCCESS	0.7
INIT-REQUEST-RESPONSE-ERROR	0.15
INIT-REQUEST-RESPONSE-DEFFERED	0.07
INT-REQUEST	0.05
INIT	0.03

Реконструируем транзакции средствами R

```r
# Маркируем транзакции
event_dt <- as.data.table(log_tbl) %>%
  # временнУю сортировку от прошлого к будущему сделаем один раз для всех
  # timeindex уже содержит миллисекунды
  setorder(store, pos, timeindex) %>%
  # все транзакции начинаются с INIT
  .[, start := (state == "INIT")] %>%
  # быстрый session_id
  .[, session_id := cumsum(start)]
```

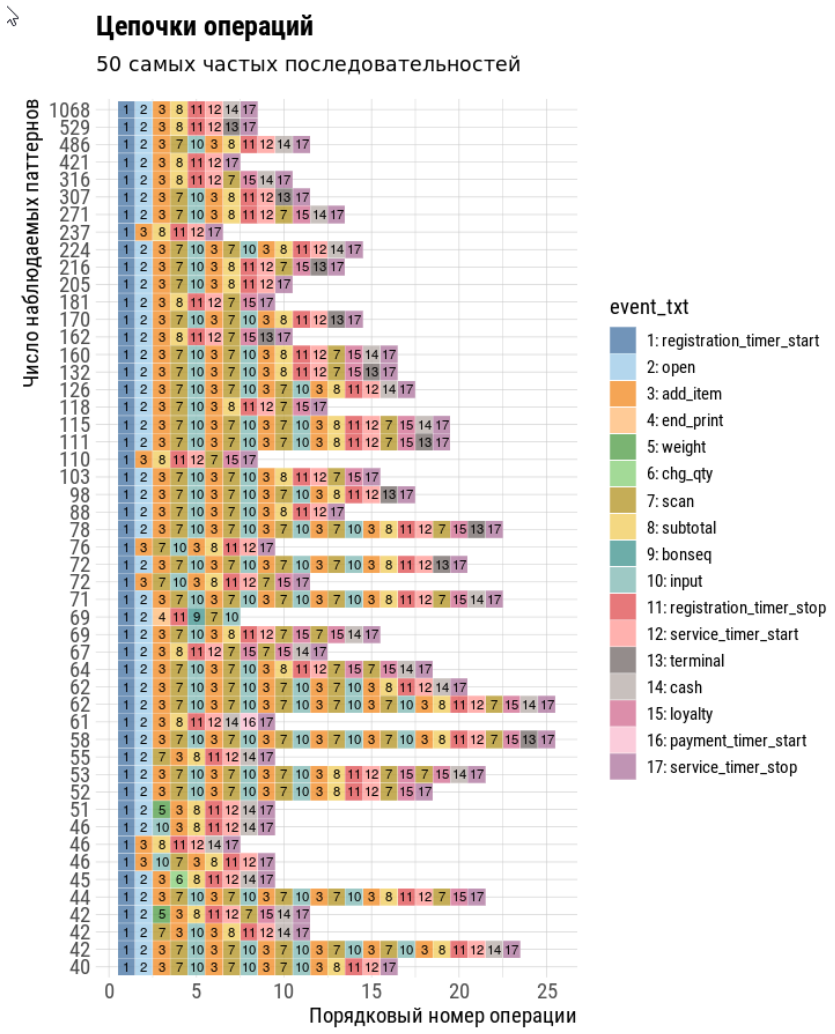
store	pos	timeindex	state	start	session_id
1001	1	2019-03-11 08:00:06	INIT	TRUE	190311_1001_1_1
1001	1	2019-03-11 08:00:07	REQUEST	FALSE	193311_1001_1_1
1001	1	2019-03-11 08:00:10	RESPONSE	FALSE	193311_1001_1_1
1001	1	2019-03-11 08:00:14	ERROR	FALSE	193311_1001_1_1
1001	1	2019-03-11 08:00:48	INIT	TRUE	193311_1001_1_2
1001	1	2019-03-11 08:00:48	REQUEST	FALSE	193311_1001_1_2
1001	1	2019-03-11 08:00:49	RESPONSE	FALSE	193311_1001_1_2
1001	1	2019-03-11 08:00:49	ERROR	FALSE	193311_1001_1_2
1001	1	2019-03-11 08:01:28	INIT	TRUE	193311_1001_1_3
1001	1	2019-03-11 08:01:30	REQUEST	FALSE	193311_1001_1_3
1001	1	2019-03-11 08:01:33	RESPONSE	FALSE	193311_1001_1_3
1001	1	2019-03-11 08:01:37	ERROR	FALSE	193311_1001_1_3
1001	1	2019-03-11 08:01:58	INIT	TRUE	193311_1001_1_4
1001	1	2019-03-11 08:01:59	REQUEST	FALSE	193311_1001_1_4
1001	1	2019-03-11 08:02:02	RESPONSE	FALSE	193311_1001_1_4
1001	1	2019-03-11 08:02:06	ERROR	FALSE	193311_1001_1_4
1001	1	2019-03-11 08:02:15	INIT	TRUE	193311_1001_1_5
1001	1	2019-03-11 08:02:18	REQUEST	FALSE	193311_1001_1_5
1001	1	2019-03-11 08:02:20	RESPONSE	FALSE	193311_1001_1_5
1001	1	2019-03-11 08:02:24	ERROR	FALSE	193311_1001_1_5
1001	1	2019-03-11 08:02:26	INIT	TRUE	193311_1001_1_6
1001	1	2019-03-11 08:02:27	REQUEST	FALSE	193311_1001_1_6
1001	1	2019-03-11 08:02:28	RESPONSE	FALSE	193311_1001_1_6
1001	1	2019-03-11 08:02:30	ERROR	FALSE	193311_1001_1_6
1001	1	2019-03-11 08:03:00	INIT	TRUE	193311_1001_1_7
1001	1	2019-03-11 08:03:05	REQUEST	FALSE	193311_1001_1_7
1001	1	2019-03-11 08:03:11	RESPONSE	FALSE	193311_1001_1_7
1001	1	2019-03-11 08:03:17	ERROR	FALSE	193311_1001_1_7
1001	1	2019-03-11 08:03:27	INIT	TRUE	193311_1001_1_8
1001	1	2019-03-11 08:03:28	REQUEST	FALSE	193311_1001_1_8

В реальной жизни все несколько сложнее



Данные имеют разрывы и потери. Маркеры не всегда присутствуют в полном объеме. События по разным чекам на одной кассе могут перекрываться по времени, большое количество событий в чеках, множественные повторы цепочек событий, миллисекундного разрешения иногда оказывается недостаточно.

Что можно использовать?



- Расщепляем транзакции на неделимые блоки
- Ищем дополнительные маркеры
- Ищем паттерны и делаем свертку циклов
- Рассматриваем метрики в разрезе «маршрутов»
- Делать реконструкции «от хвоста»
- …

Реконструируем транзакции средствами R

```r
# Маркируем транзакции
event_dt <- as.data.table(log_tbl) %>%
  # временнУю сортировку от прошлого к будущему сделаем один раз для всех
  # timeindex уже содержит миллисекунды
  setorder(store, pos, timeindex) %>%
  # все транзакции начинаются с INIT
  .[, start := (state == "INIT")] %>%
  # быстрый session_id
  .[, session_id := cumsum(start)] %>%
  # в отсортированном списке событий протягиваем куммулятивный максимум времени, ориентируясь на INIT
  .[, time_locf := cummax(timeindex * as.numeric(start)), by = .(store, pos)] %>%
  .[, time_shift := timeindex - time_locf] %>%
  # маркируем транзакционное окно, при генерации мы его установили равным 30 секунд
  .[, lost_chain := time_shift > 30]
```

Паттерн операции един для всех событийных записей по этой транзакции, поэтому оптимально работать на уровне словаря. Не стесняемся применения регулярных выражений

```r
# Соберем паттерны операций
bp_dict <-  clean_dt %>%
  .[, .(bp_pattern = stri_join(event, collapse = "-")), by = session_id]

# Раскладываем последовательность событий по паттернам
df <- data.table::copy(bp_dict) %>%
  # погасим все события после 'registration_timer_stop'
  .[, bp_pattern := re2r::re2_replace_all(bp_pattern, "(registration_timer_stop).*$", "\\1")] %>%
  # свертка `add_item-scan-input` без учета количества
  .[, bp_pattern := re2r::re2_replace_all(bp_pattern, "((add_item-scan-input-)+)", "@add-")] %>%
  .[, bp_pattern := re2r::re2_replace_all(bp_pattern, "(subtotal-registration_timer_stop-
service_timer_start)", "@reg_stop")]
```

Улучшаем отображение процесса

Прямой подход

```r
# Сделаем визуализацию бизнес процессов
# Прямой подход
eventlog_tbl <- as_tibble(clean_dt) %>%
  mutate(activity_instance_id = dplyr::row_number()) %>%
  bupaR::simple_eventlog(case_id = "session_id",
                         activity_id = "event",
                         timestamp = "timestamp")

graph <- eventlog_tbl %>%
  edeaR::filter_activity_frequency(percentage = 0.8) %>%
  edeaR::filter_trace_frequency(percentage = 0.7) %>%
  processmapR::process_map(type_nodes = processmapR::frequency("relative_case"),
                           # type_edges = processmapR::performance(median, "secs"),
                           type_edges = processmapR::frequency("relative_case"),
                           render = FALSE)
```

Ручной контроль

```r
# Вариант чистки после сборки process map
receipt_process_map <- eventlog_tbl %>%
  processmapR::process_map(type_nodes = processmapR::frequency("relative_case"),
                           type_edges = processmapR::frequency("relative_case"),
                           sec_edges = processmapR::performance(median, "secs",
                                                                flow_time = "inter_start_time"),
                           # rankdir = "TB",
                           render = FALSE)

edges_to_remove <- receipt_process_map %>%
  DiagrammeR::get_edge_df() %>%
  as_tibble() %>%
  mutate(percent = parse_number(label)) %>%
  # фильтруем в процентах (5 = 5%)
  filter(percent < 5) %>%
  pull(id)

filtered_graph <- receipt_process_map %>%
  DiagrammeR::select_edges_by_edge_id(edges = edges_to_remove) %>%
  DiagrammeR::delete_edges_ws() %>%
  DiagrammeR::select_nodes_by_degree("deg == 0") %>%
  DiagrammeR::delete_nodes_ws()

# DiagrammeR::render_graph(filtered_graph)
DiagrammeR::export_graph(filtered_graph, file_name = "bp3.png", file_type = "png",
                         title = NULL, width = 6000, height =  NULL)
```

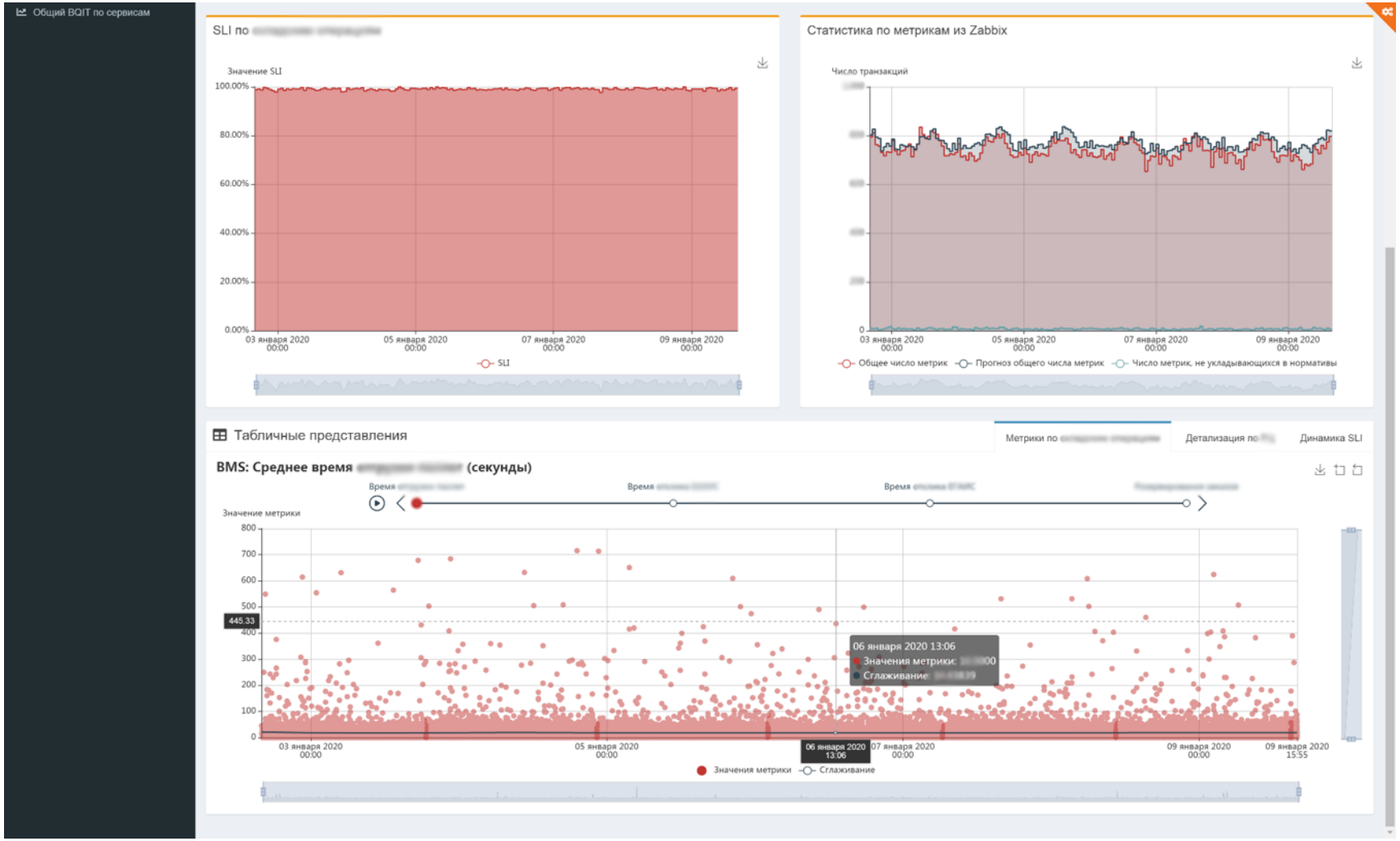
Полезные трюки

- Для основного процессинга только data.table (скорость, работа по ссылкам) + учет внутренней оптимизации запросов
- POSIXct может содержать миллисекунды (хоть штатно и не отображается, но можно подправить с помощью options(digits.secs=X)), прячем их туда, проще будет сравнивать и сортировать
- Тщательно вспоминаем про базовые типы данных и принципы их хранения + ITime, IDate, эмуляция integer64
- Избегаем физической сортировки внутри групп! Однократная физическая сортировка всего вектора гарантирует сортировку данных в группировках
- Избегаем вычислений внутри групп! Все что можно, стараемся выполнить на исходных данных (применяем векторизацию, снижаем накладные на вызовы функций)
- Для слияния атрибутов не используем “summary” функции по группам, проводим пакетное слияние с помощью join
- Используем таймаут на транзакции, чтобы бороться с временными разрывами
- Методы locf (Last Observation Carried Forward) работают медленно. Для переноса свойств по шкале времени используем cumsum, cummax
- Не стесняемся использовать регулярные выражения для анализа паттернов транзакций и бизнес-операций
- Трудоемкие операции, такие как преобразование POSIX -> строка, поиск по регуляркам и т.п. делаем не поэлементно, а на свертках. Накладные на внутреннюю индексацию и группировку преобразуемого поля несопоставимо меньше
- Активно используем многопоточность (в т.ч. внутрипакетную)
- Не пренебрегаем микрооптимизацией. Например, stri_c в несколько раз быстрее paste0
- Читаем NEWS по новым релизам используемых пакетов, используем новые фичи

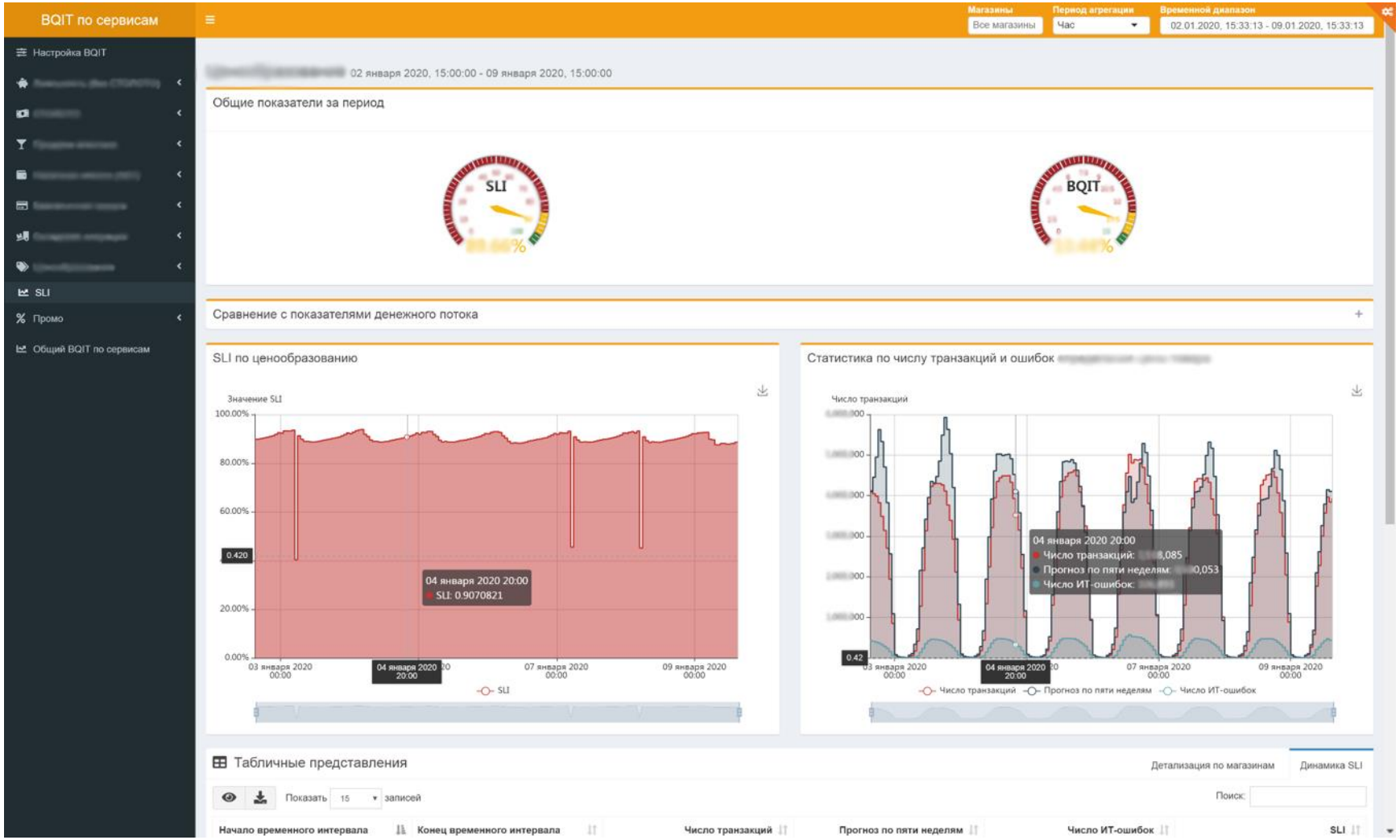
Ссылки на развернутые публикации

- Бизнес-процессы в enterprise компаниях: домыслы и реальность. Проливаем свет с помощью R (https://habr.com/ru/post/461463/)
- Несколько штрихов о работе с идентификаторами bigint в R (https://habr.com/ru/post/466829/)

BQIT App


Общий BQIT по сервисам
SLI по
Значение SLI
100.00%
80.00%
60.00%
40.00%
20.00%
0.00%
03 января 2020 00:00
05 января 2020 00:00
07 января 2020 00:00
09 января 2020 00:00
SLI
Статистика по метрикам из Zabbix
Число транзакций
Общее число метрик
Прогноз общего числа метрик
Число метрик, не укладывающихся в нормативы
Табличные представления
Метрики по
Детализация по
Динамика SLI
BMS: Среднее время (секунды)
Время
Значение метрики
800
700
600
500
445.33
400
300
200
100
0
06 января 2020 13:06
Значения метрики:
Сглаживание
06 января 2020 13:06
09 января 2020 15:55
Значения метрики
Сглаживание

BQIT App


BQIT по сервисам
Настройка BQIT
SLI
Промо
Общий BQIT по сервисам
Магазины
Все магазины
Период агрегации
Час
Временной диапазон
02.01.2020, 15:33:13 - 09.01.2020, 15:33:13
02 января 2020, 15:00:00 - 09 января 2020, 15:00:00
Общие показатели за период
SLI
BQIT
Сравнение с показателями денежного потока
SLI по ценообразованию
Значение SLI
100.00%
80.00%
60.00%
0.420
20.00%
0.00%
04 января 2020 20:00
SLI: 0.9070821
03 января 2020 00:00
04 января 2020 20:00
07 января 2020 00:00
09 января 2020 00:00
SLI
Статистика по числу транзакций и ошибок
Число транзакций
04 января 2020 20:00
Число транзакций:
Прогноз по пяти неделям:
Число ИТ-ошибок:
0.42
Число транзакций
Прогноз по пяти неделям
Число ИТ-ошибок
Табличные представления
Детализация по магазинам
Динамика SLI
Показать 15 записей
Поиск:
Начало временного интервала
Конец временного интервала
Число транзакций
Прогноз по пяти неделям
Число ИТ-ошибок
SLI

Область применения BMS гораздо шире

Что реализовано, но осталось за кадром тематики process mining

- Контроль качества работы обновленного ПО
- Мониторинг аномалий в обслуживании
- Проведение A\B тестирования нового оборудования\новых прошивок
- Мониторинг некорректных конфигураций\логики работы ИТ, значимых с точки зрения штрафов
- Анализ продаж алкоголя (РЦ-магазин-ЕГАИС-POS)
- Анализ и выявление системных проблем с продажей продукции
- Поведенческий анализ кассиров, формирование рекомендаций их целевого дообучения
- Анализ эффективности работы директоров магазина
- Прогнозирование очередей на кассах
- Расчет недобора за счет сбоя в работе оборудования. Построение претензионной работы с поставщиками ИТ услуг (например, система лояльности, провайдеры каналов связи)
- Контур обратной связи (оркестратор)
- Аналитика по программам лояльности (включая геоаналитику и глубину проникновения)
- Детектирование фродовых операций с картами лояльности
- Расчет различных производных KPI для бизнес-подразделений