Рабочая документация
(в рамках стратегии импортозамещения)

Автоматизированная Система Управления Железнодорожными Перевозками

(АСУ ЖДП)

Модификации программного обеспечения «Автоматизированная система управления перевозками АО «ЯЖДК» в рамках перевода на свободное ПО

Руководство администратора

Москва
2024

АННОТАЦИЯ

Настоящее Руководство администратора разработано по теме: «Модификации программного обеспечения «Автоматизированная система управления перевозками АО «ЯЖДК» в рамках перевода на свободное ПО» с целью описания механизмов работы в системе АСУ ЖДП в рамках стратегии импортозамещения.

Документ описывает порядок действий по обеспечению технологического процесса функционирования БД и действий при его нарушении, порядок копирования, восстановления и защиты данных от разрушений и несанкционированного доступа. Документ включает как непосредственно Руководство администратора, так и описание комплекса программ АСУ ЖДП ВЕБ. Информация по развертыванию сайта АСУ ЖДП ВЕБ приведена в Руководстве пользователя по теме: «Модификации программного обеспечения «Автоматизированная система управления перевозками АО «ЯЖДК» в рамках перевода на свободное ПО».

Документ входит в состав документации по Автоматизированной Системе Управления Железнодорожными Перевозками (АСУ ЖДП).

В процессе эксплуатации АСУ ЖДП возможно внесение изменений в настоящий Документ. Изменения должны быть согласованы со всеми заинтересованными сторонами: АО «Ямальская железнодорожная компания», ФЗ (при наличии функций, описанных в п.2.1 данного документа), организацией – разработчиком.

Все изменения настоящего документа подлежат регистрации в Листе регистрации изменений.

Процедура переподписания настоящего Документа (титульный лист и таблица Согласования) осуществляется при модификации в целом АС (при сдаче/приемке новой очереди).

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Настоящее руководство разработано для администраторов и программистов, выполняющих сопровождение системы работы «Автоматизированной системы управления перевозками АО «ЯЖДК» на базе свободного ПО, далее АСУ ЖДП ВЕБ.

1. РУКОВОДСТВО АДМИНИСТРАТОРА

1.1 ФУНКЦИИ АДМИНИСТРИРОВАНИЯ

Инсталляция и подготовка системы к эксплуатации:

- Установка фреймворка.Net (на момент написания инструкции 17.07.24 версия 8.0)
- Установка и настройка сервера приложений Nginx
- Установка оболочки сайта.
- Установка компонентов сайта
- Демонизация сайта (настройка служб бесперебойной работы)
- Настройка прав пользователя
- Настройка абонента Телеобработки сайта
- Настройка абонента Телеобработки клиента

Обновление компонентов системы

Слежение за работой сайта

1.2 ИНСТАЛЛЯЦИЯ И ПОДГОТОВКА СИСТЕМЫ К ЭКСПЛУАТАЦИИ

Методика установки фреймворка и сервера приложений Nginx с подключением к Интернету подробно описана в документе Развёртывание сайта АСУ ЖДП ВЕБ.docx

В случае если операционная система имеет уровень защищённости, не допускающий подключение к Интернету необходимо скачать дистрибутивы из интернета, перенести их на целевую машину и установить. Установка .Net 7.0 подробно описана в документе Развёртывание сайта АСУ ЖДП ВЕБ на АСТРА Воронеж.docx. Предполагается, что процесс установки .Net 8.0 мало отличается от описанного в документе. Необходимо только иметь в виду повышение версии дистрибутивов. Но, поскольку фактическая установка не проводилась из-за отсутствия доступа к таким системам, а в работе с Linux возможны нюансы, решено ограничиться только рекомендациями по .Net 7.0

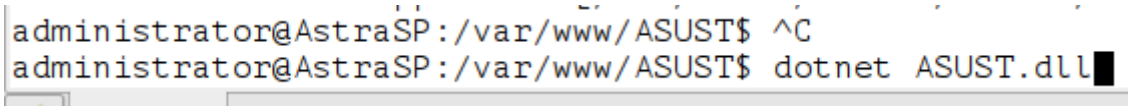
1.3 ОБНОВЛЕНИЕ КОМПОНЕНТОВ СИСТЕМЫ

На момент написания документа порядок обновления не разработан. Пока обновление идёт копированием новых компонентов в папки на сервере. Если после обновления компонента сайт не запускается службой, необходимо перейти в Терминале в папку сайта:

и попытаться запустить сайт вручную командой:

dotnet ASUST.dll

(чувствительна к регистру!)



Если запуск происходит с ошибкой и выявлен компонент, мешающий работе, то отключить компонент до выяснения причин можно в конфигурационном файле Setting.json, закомментировав соответствующий компоненту раздел.

```
{
    «Module»: «ARM_TK.dll», //Имя библиотеки
    «Name»: «ТК», //Видимое в меню имя страницы
    «Path»: «Pages\\ARM_TK», //Относительный путь к странице
    «Area»: «ARM_TK», //Наименование области
    «Namespace»: «cit.ARM_TK», //Пространство имён
    «Title»: «Оператор технической конторы», //Видимое в заголовке сайта имя страницы
    «OperIDs»: [ 900, 901 ] //Идентификаторы операции, позволяющей показывать страницу
 },
 /*{
    «Module»: «ARM_RS.dll», //Имя библиотеки
    «Name»: «РС», //Видимое в меню имя страницы
    «Path»: «Pages\\ARM_RS», //Относительный путь к странице
    «Area»: «ARM_RS», //Наименование области
    «Namespace»: «cit.ARM_RS», //Пространство имён
    «Title»: «АРМ руководителя станции», //Видимое в заголовке сайта имя страницы
    «OperIDs»: [ 1800 ] //Идентификаторы операции, позволяющей показывать страницу
 },*/
        {
```

В приведённом примере компонент АРМ ТК загружается, а компонент АРМ РС – игнорируется

1.4 СЛЕЖЕНИЕ ЗА РАБОТОЙ САЙТА

Для удобства администрирования на сайте разработан комплекс сервисов (рисунок 1):

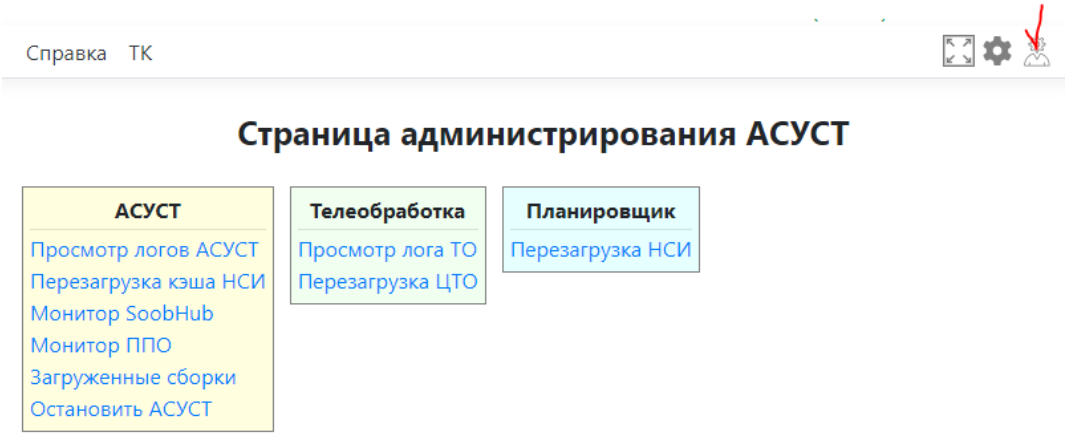


Рисунок 1

Чтобы получить доступ к сервисам администрирования пользователю необходимо входить в группу, привязанной к роли Администратор АРМ АСУ ЖДП WEB (RoleID=8). В этом случае в правом верхнем углу появится иконка «Администрирование» с переходом на главную страницу администрирования. В этом разделе рассматривается работа сервисов.

1.4.1 ПРОСМОТР ЛОГОВ АСУ ЖДП

Все логи работы оболочки и компонентов можно посмотреть в этом разделе (рисунок 2).

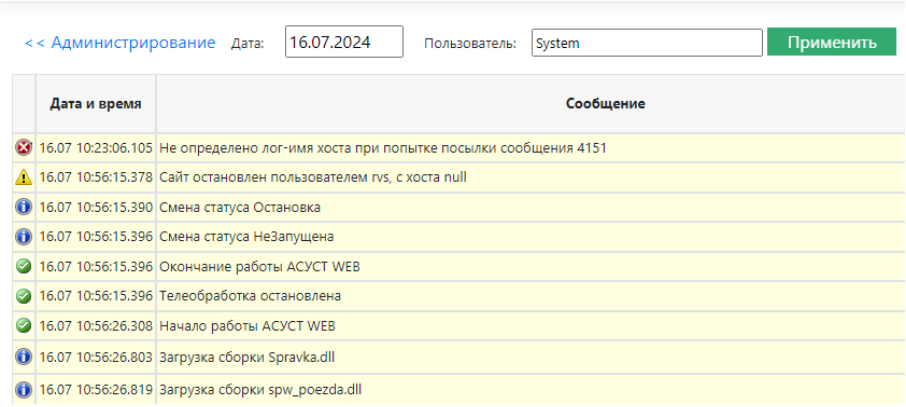


Рисунок 2

Для загрузки логов необходимо выбрать дату и имя пользователя от которого пишется лог и нажать кнопку «Применить». По умолчанию используется пользователь System. В этот раздел попадает информация, если разработчик, проектируя лог, не указал имя пользователя. В противном случае, в лог попадает информация по работе заданного пользователя (рисунок 3):

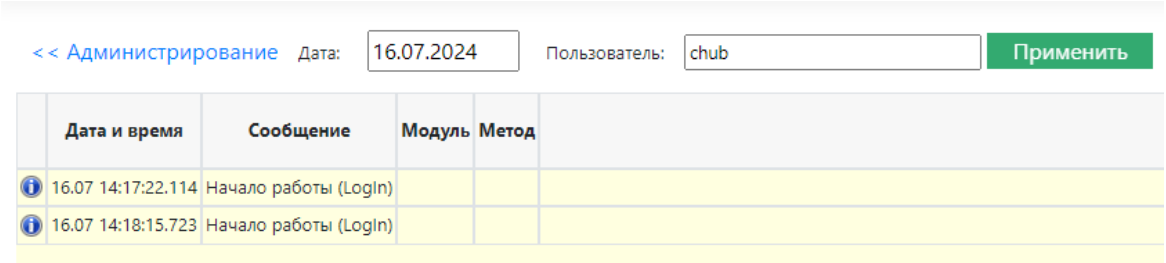


Рисунок 3

Если за заданную дату записей нет система выдаст ошибку (рисунок 4):

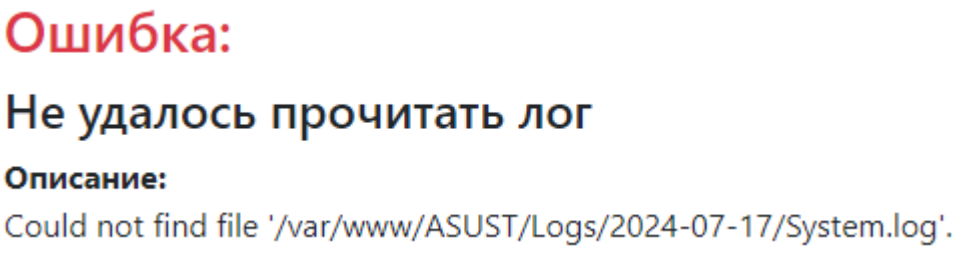


Рисунок 4

Чтобы скопировать подробности сообщения необходимо правой кнопкой щёлкнуть по строке сообщения и выбрать «Копировать подробности» в контекстном меню (рисунок 5).

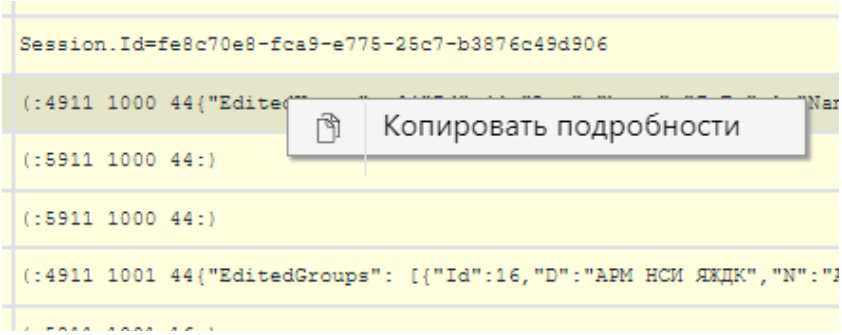


Рисунок 5

1.4.2 ПЕРЕЗАПУСК КЭША НСИ

Нормативно справочная информация (НСИ) считывается из базы данных (далее БД) в момент обращения и хранится в кэше НСИ, откуда берётся по запросу без обращения к БД. Чтобы не перезапускать сайт после изменений в таблице НСИ в БД, достаточно очистить её кэш. Тогда при обращении обновлённая таблица закачается из БД. Для обнуления таблицы на странице необходимо отметить нужные таблицы и нажать «Перезагрузить» (рисунок 6).

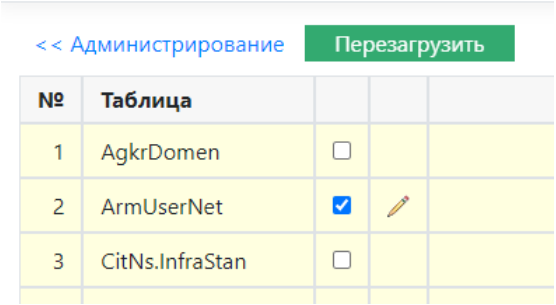
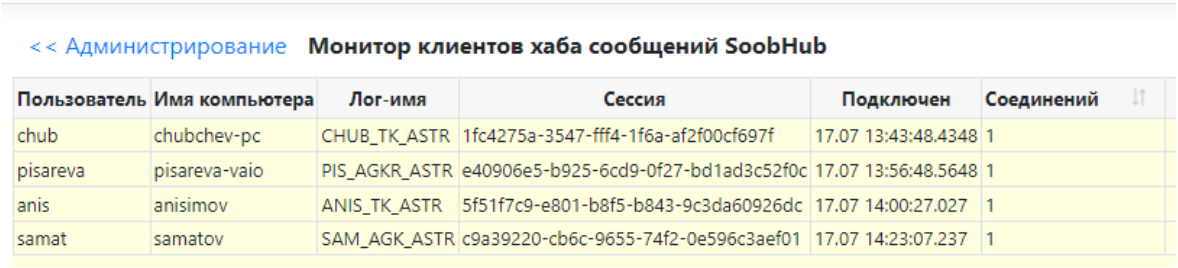


Рисунок 6

1.4.3 МОНИТОР SOOBHUB

Двусторонние соединения сайта (сервера) и браузера (клиента) осуществляется через веб-сокеты. Если сообщение на сервер приходит для

какого-то определённого абонента (по лог.имени), то оно посылается по всем открытым соединениям этого лог-имени. Для мониторинга этих сообщений разработан сервис Монитор SoobHub (рисунок 7).



<< Администрирование **Монитор клиентов хаба сообщений SoobHub**

Пользователь	Имя компьютера	Лог-имя	Сессия	Подключен	Соединений
chub	chubchev-pc	CHUB_TK_ASTR	1fc4275a-3547-fff4-1f6a-af2f00cf697f	17.07 13:43:48.4348	1
pisareva	pisareva-vaio	PIS_AGKR_ASTR	e40906e5-b925-6cd9-0f27-bd1ad3c52f0c	17.07 13:56:48.5648	1
anis	anisimov	ANIS_TK_ASTR	5f51f7c9-e801-b8f5-b843-9c3da60926dc	17.07 14:00:27.027	1
samat	samatov	SAM_AGK_ASTR	c9a39220-cb6c-9655-74f2-0e596c3aef01	17.07 14:23:07.237	1

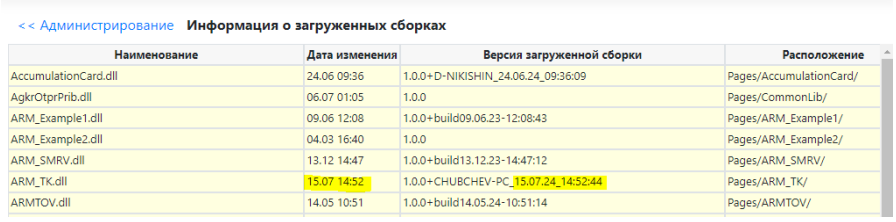
Рисунок 7

1.4.4 МОНИТОР ППО

Для работы с сообщениями подсистемы поездообразования и планирования маневровой работы (ППО) организован отдельный сервис мониторинга соединений аналогично п. 1.4.3.

1.4.5 ЗАГРУЖЕННЫЕ СБОРКИ

Этот сервис служит для мониторинга версий загруженных сборок (рисунок 8).



<< Администрирование **Информация о загруженных сборках**

Наименование	Дата изменения	Версия загруженной сборки	Расположение
AccumulationCard.dll	24.06 09:36	1.0.0+D-NIKISHIN_24.06.24_09:36:09	Pages/AccumulationCard/
AgkrOtprPrib.dll	06.07 01:05	1.0.0	Pages/CommonLib/
ARM_Example1.dll	09.06 12:08	1.0.0+build09.06.23-12:08:43	Pages/ARM_Example1/
ARM_Example2.dll	04.03 16:40	1.0.0	Pages/ARM_Example2/
ARM_SMRV.dll	13.12 14:47	1.0.0+build13.12.23-14:47:12	Pages/ARM_SMRV/
ARM_TK.dll	15.07 14:52	1.0.0+CHUBCHEV-PC_15.07.24_14:52:44	Pages/ARM_TK/
ARMTOV.dll	14.05 10:51	1.0.0+build14.05.24-10:51:14	Pages/ARMTOV/

Рисунок 8

Сборка может быть скопирована, на сервер, но, если не перезагрузить сайт, то загруженной останется старая сборка. Поэтому необходимо следить за датами. В столбце «Дата изменения» показывается дата изменения файла,

который лежит в файловой системе. В столбце «Версия…» показывается в том числе дата загруженной в оперативную память сборки (если разработчик сборки проделал для этого определённые шаги)

1.4.6 ПРОСМОТР ЛОГА ТО

На этой странице можно увидеть лог Телеобработки сайта (рисунок 9)

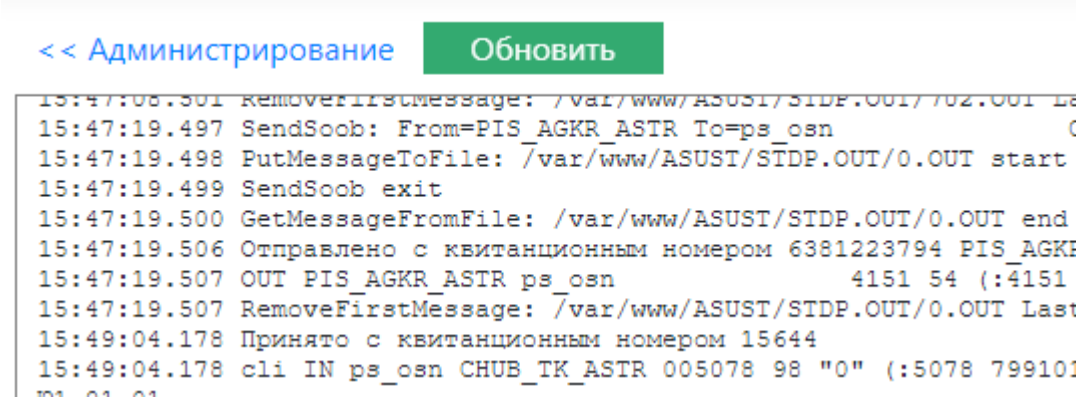


Рисунок 9

1.4.7 ПЕРЕЗАГРУЗКА ЦТО

На этой странице (рисунок 10) можно послать сообщение на перезагрузку центральной Телеобработки. Например, при добавлении нового абонента.

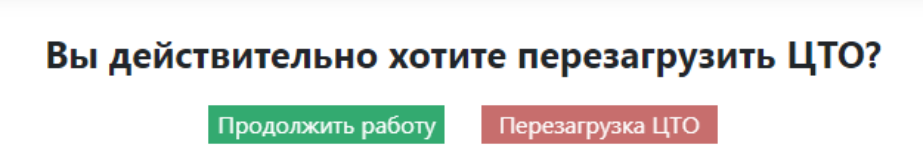


Рисунок 10

1.4.8 ПЕРЕЗАГРУЗКА НСИ ПЛАНИРОВЩИКА

На этой странице можно послать сообщение на перезагрузку НСИ Планировщика, без его выгрузки.

2. ОПИСАНИЕ КОМПЛЕКСА ПРОГРАММ АСУ ЖДП ВЕБ

2.1 СТРУКТУРА ПО

Комплекс программ для работы сайта АСУ ЖДП ВЕБ (далее сайт) включает в себя *оболочку* и набор *веб-АРМов*

Оболочка состоит из комплекса программных модулей (библиотек), комплекса системных библиотек, дополнительных файлов, такие как скрипты, таблицы стилей, изображения и др.

Веб-АРМы состоят из программных модулей (библиотек) и располагаются каждый в своей папке директории Pages

2.1.1 ФУНКЦИИ КОМПОНЕНТОВ ПО

Оболочка – комплекс сборок в составе АСУ ЖДП WEB обеспечивающий координацию, и общие функции Веб-АРМов, такие как авторизация, загрузка НСИ, печать, логгирование, сохранение пользовательских настроек, стиль элементов управления и др.

Веб-АРМ – комплекс сборок в составе АСУ ЖДП WEB обеспечивающий функционал работы одного АРМа (АРМ ТК, АРМ ТОВ, Справочная система и т.д.)

2.1.2 СРЕДСТВА РАЗРАБОТКИ

Сайт разработан в IDE Visual Studio 2022 на языках C# и JavaScript

2.1.3 ПОЛЬЗОВАТЕЛЬСКИЙ ИНТЕРФЕЙС

Сайт доступен пользователю в браузере. В качестве основных в рамках свободного ПО рекомендуются Яндекс.Браузер версии не ниже 21.9.1.684 и Mozilla Firefox версии не ниже 93.0. Google Chrome также тестировался в качестве средства просмотра сайта. На других браузерах ПО не тестировалось.

2.1.4 ОПЕРАЦИОННАЯ СИСТЕМА

Операционная система: Linux Debian 10. В качестве тестовых полигонов использовались Astra Linux (Orel) 2.12.22, а также Astra Linux Special Edition "Воронеж"

Приемлема также Windows не ниже версии 10,

Сайт для работы требует установки фреймворка .Net 8.0

Сервер приложений: Nginx версии не ниже 1.14.1

2.1.5 РАЗРАБОТЧИКИ

Оболочка: Чубчев.

Веб-АРМы:

- АРМ Технической конторы: Чубчев, Фатеев, Романов, Логунов
- Справочная система: Чубчев
- Приёмосдатчик: Суворов
- АРМ НСИ: Бородий

2.2 КОМПОНЕНТЫ

2.2.1 ОБОЛОЧКА

2.2.1.1 Разработчик: Чубчев Г.Н.

2.2.1.2 Назначение: обеспечивает координацию, и общие функции Веб-АРМов, такие как авторизация, загрузка НСИ, печать, логгирование, сохранение пользовательских настроек, стиль элементов управления и др

2.2.1.3 Программные компоненты:

appsettings.json

ASUST.deps.json

ASUST.dll

ASUST.runtimeconfig.json

ASUST.Views.dll

ASUSTCommon.dll

ObrClsUserNastr.dll

ObrNetASUSTUserNastr.dll

ServNet.dll

Setting.json

2.2.1.4 Системные компоненты:

Dapper.dll

Microsoft.AspNetCore.JsonPatch.dll

Microsoft.AspNetCore.Mvc.Razor.Extensions.dll

Microsoft.AspNetCore.Mvc.Razor.Host.dll

Microsoft.AspNetCore.Mvc.Razor.RuntimeCompilation.dll

Microsoft.AspNetCore.Razor.Language.dll

Microsoft.CodeAnalysis.CSharp.dll

Microsoft.CodeAnalysis.dll

Microsoft.CodeAnalysis.Razor.dll

Microsoft.Extensions.DependencyModel.dll

Microsoft.Extensions.FileProviders.Embedded.dll

Microsoft.Extensions.PlatformAbstractions.dll

Mvc.Grid.Core.dll

NetTOCore.dll

Newtonsoft.Json.Bson.dll

Newtonsoft.Json.dll

Npgsql.dll

, а также папка refs со всеми системными файлами.

2.2.1.5 Дополнительные файлы:

Папка wwwroots

2.2.2 АРМ ТК

2.2.2.1 Назначение: обеспечивает функционал работы АРМ технической конторы.

2.2.2.2 Программные компоненты:

Наименование	**Разработчики**	**Расположение**
ARM_TK.dll	Чубчев Г.Н. Фатеев С.Р., Романов А.Ю.	/Pages/ARM_TK
LK.dll	Чубчев Г.Н. Карнаухова Г.Ю	/Pages/ARM_TK

2.2.3 СПРАВОЧНАЯ СИСТЕМА

2.2.3.1 Назначение: обеспечивает получение различной справочной информации

2.2.3.2 Программные компоненты:

Наименование	Разработчики	Расположение
Spravka.dll	Чубчев Г.Н.	/Pages/ Spravka
spw_arhiv.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_arhiv
spw_doc.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_doc
spw_mestopod.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_mestopod
spw_nar.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_nar
spw_park.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_park
spw_poezda.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_poezda
spw_prognoz.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_prognoz
spw_put.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_put
spw_stat.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_stat
spw_tov.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_tov
spw_vagon.dll	Чубчев Г.Н.	/Pages/ Spravka/Pages/spw_vagon

3. АРХИТЕКТУРА АСУ ЖДП НА СВОБОДНОМ ПО

Архитектурная схема АСУ ЖДП приведена на рисунке 11.

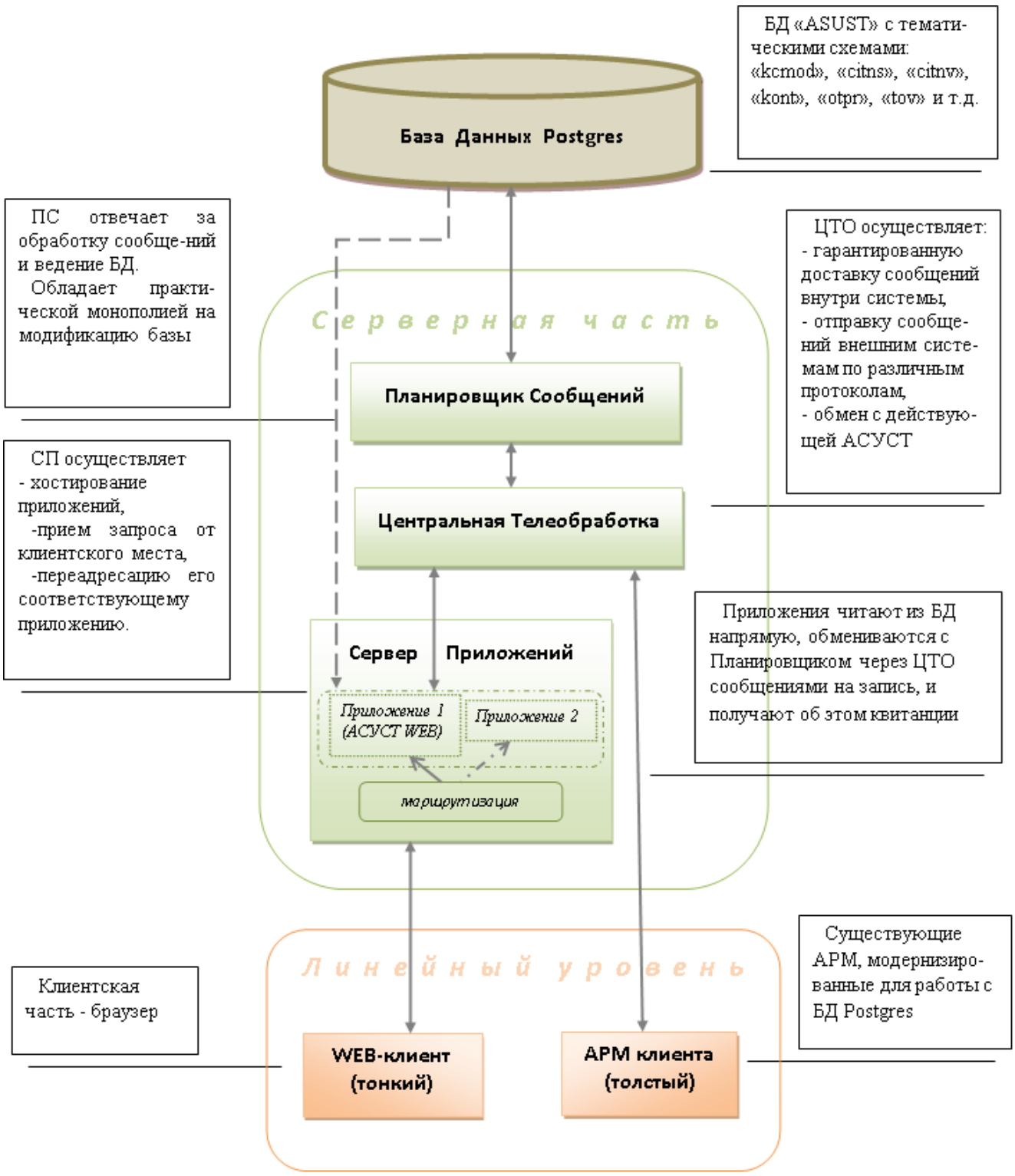


Рисунок 11

АСУ ЖДП использует архитектуру с "тонким" клиентом. Клиентская часть - браузер.

Браузер обращается к приложениям, размещённым на Сервере Приложений.

3.1 СЕРВЕРНАЯ ЧАСТЬ

Серверная часть состоит из:

- Сервер Приложений, отвечающий за работу с клиентскими местами, осуществляет хостирование приложений: поддержку их в работающем состоянии, масштабирование и т.д. Принимает запросы от клиентских мест, переадресует их приложениям. Организует взаимодействие между приложениями, планировщиком и БД.

- Приложения – программные модули, размещённые на Сервере приложений, формирующие интерфейс взаимодействия с пользователем (веб-страницы), реализующие бизнес-логику. Принимает запросы от WEB АРМ на чтение и запись в/из БД. Читает из БД приложение, для записи информации в БД, формирует сообщение и посылает через ЦТО Планировщику.

- Планировщика системы, отвечающего за ведение БД

- ЦТО, Центральная Телеобработка, отвечающая за доставку сообщений.

3.2 БАЗА ДАННЫХ

В качестве БД используется PostgrePro.

3.3 ТЕХНОЛОГИИ МИГРАЦИИ СУЩЕСТВУЮЩЕЙ АСУ ЖДП В НОВУЮ СИСТЕМУ

Невозможно мгновенно отключить старую систему и подключить новую. Значит необходима технология и функционал, обеспечивающие

параллельное функционирование двух систем и плавную передачу функций из старой в системы в новую.

- первоначальная закачка БД

- система оперативных перезапросов на уточнение данных из новой систему в старую

- реализация варианта параллельной работы двух систем, причем часть АРМ подключена к старому серверу, а часть к новому. Переход на работу в новой системе должен происходить постанционно.

- возможность параллельной работы разных версий АРМ одного типа, например АРМ ТК, в новой системе.

- постепенная передача функций от старой системы к новой.

3.4 ФУНКЦИИ ЧАСТЕЙ ПО

3.4.1 БД POSTGRESPRO

БД PostgresPro используется для создания объектов БД АСУ ЖДП, таких как схемы, таблицы, хранимые процедуры. Таблицы используется для хранения данных модели и Нормативно-Справочной информации.

3.4.1.1 Информационно-логическая схема Базы Данных

Информационно-логическая схема Базы Данных представлена на рисунке 12.

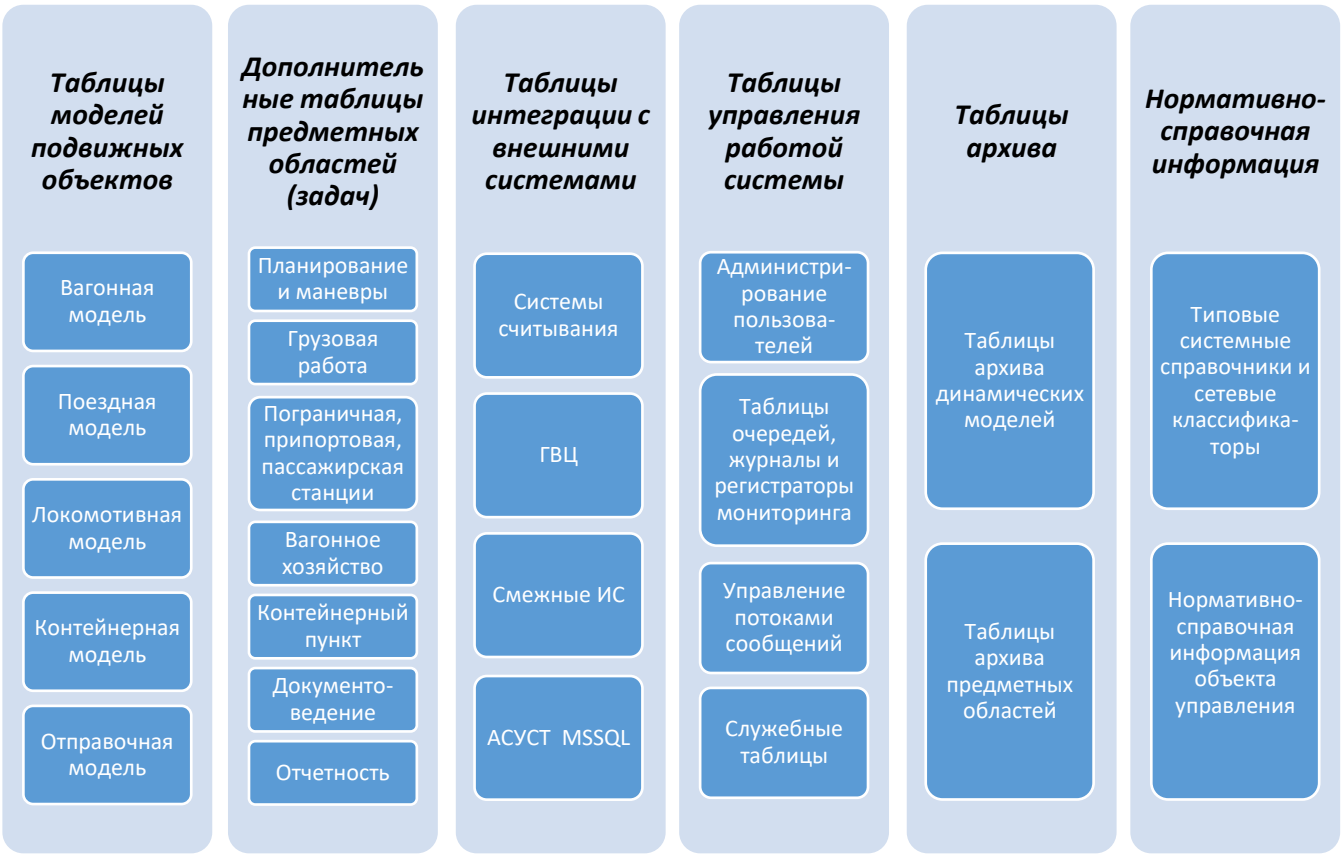


Рисунок 12

3.4.1.2 Описание внутренней архитектуры (схем хранения)

База данных одна, называется ASUST. Все данные собраны в одной БД, т.к. невозможно работать в одном соединении более чем с одной БД.

Внутри БД деление по схемам:

Имя	Комментарий
arx2011	Глобальный архив за 20 год, ноябрь месяц
arx2012	Глобальный архив за 20 год, декабрь месяц
arx2101	Глобальный архив за 21 год, январь месяц
Citbd	Новые таблицы модели, дополнение к kcmod
Citns	Таблицы классификаторов, постоянное НСИ
Citnv	Переменное НСИ
Fdw	Таблицы, данные которых связанны с таблицами в БД

	MSSQL
Kcmod	Таблицы вагонной и поездной модели, ряд других таблиц модели
Kont	Контейнерная модель
Logs	Служебные таблицы вагонников
Mest	Местная работа
Otpr	Отправочная модель
Tov	Модель вагонников
Tovarx	Архив вагонников

3.4.1.3 Таблица событий

В АСУ ЖДП все технологические события оформляются в виде сообщений (входная информация). Создана таблица событий, где в рамках обработки сообщения создается запись о событии со своим идентификатором. Все остальные таблицы динамических объектов, архивов и журналов содержат ссылку EventId на событие.

Присвоенное значение EventId с основного СП передается на резервный сервер вместе с сообщением, что обеспечивает синхронное ведение баз данных на обоих серверах.

Структура таблицы событий (Event):

Имя	Комментарий
EventId	Идентификатор события
date_pop	Технологическое время события
date_sys	Системное время события
kod_oper	Код события
kod_operUt	Уточняющий код
InfraStanId	Идентификатор станции, на которой произошло событие

ArmId	Источник события
UserId	Автор события(пользователь)
SoobId	Ссылка на сообщение ТО

Параллельно с Event создается таблица EventArx:

Имя	Комментарий
EventId	Идентификатор события
date_pop	Еехнологическое время события
date_sys	Системное время события
InfraStanId	Идентификатор станции, на которой произошло событие
kod_oper	Код события
kod_operUt	Уточняющий код
priznaki	Признак отмененного события, 1- событие отменено
EventOtmId	Идентификатор отмененного события, пишется в 2 записи, кто отменяет и кого отменяет
ArmId	Источник события
UserId	Автор события(пользователь)
SoobId	Ссылка на сообщение ТО

Обе таблицы поддерживаются параллельно.

Event хранит данные до тех пор, пока в модели существует динамический объект (поезд, вагон, локомотив), который содержит ссылку на соответствующее событие.

EventArx подчиняется общим правилам архивации.

Таким образом, через поле EventId связываются динамические и архивные данные во всех моделях: поездной, вагонной, локомотивной, отправочной, контейнерной.

3.4.1.4 Описание НСИ инфраструктуры

Описанная ниже система унификации объектов инфраструктуры позволяет при необходимости добавлять новые типы инфраструктуры и привязывать к ним динамические объекты без корректировки ПО.

Все объекты инфраструктуры имеют свой тип и принадлежат к одному из двух основных видов: простые и составные объекты.

Перечень типов в CitNs..TypeInfraObject:

Простые:

1 - вся сеть РЖД

3 - станция

4 - парк

2 - дорога

5 - место подачи

6 - подход к станции

7 – станционный путь

8 – локомотивное Депо

9 - район управления(отделение)

10 - грузовой район

11 - соединительный путь(кАРМан)

12 – вагонное депо

20 –страны

Составные:

102 - набор дорог

103 - набор станций

104 - набор парков

105 - набор мест подач

и т.д.

Каждый объект имеет свой уникальный ИД, сквозной в рамках всех типов.

Все простые объекты детально описаны в таблице НСИ соответствующего типа:

- CitNv..InfraDor
- CitNv..InfraStan

ит.д.

Перечень участников всех составных объектов хранится в одной таблице:

CitNv..InfraDepend

Все объекты (составные и простые) перечислены в сводной таблице объектов инфраструктуры:

CitNv..InfraObj

При создании нового объекта он в рамках одной транзакции прописывается в двух таблицах: сначала создается запись в общей таблице CitNv..InfraObj, а затем, если это простой объект, запись с таким же ObjId создается в таблице.

3.4.1.5 Описание динамических моделей

Основные технологические блоки таблиц:

- вагонный
- поездной
- локомотивный
- отправочный
- контейнерный
- местная работа
- работа с ЭДО, цифровая подпись

- вагонники (ТОВ, ПТО)
- отчетность
- пограничные станции
- портовые станции
- взаимодействие с устройствами считывания
- обслуживание транспортных компаний
- взаимодействие с ФТС
- взаимодействие с ЭТРАН
- иностранные вагоны
- иностранные поезда
- взаимодействие с действующей АСУ ЖДП на MS SQL
- авторизация и аутентификация. Администрирование пользователей.
- унифицированное описание объектов инфраструктуры
- классификаторы
- переменное НСИ
- план формирования, разметка вагонов
- картотеки связанные с внешними системами
- различные журналы и регистраторы

- **Описание вагонной модели**

Модель состоит из главной таблицы kcmod.Vagon и набора дочерних:

- vagon_mest - местная работа
- vagoncs - данные о цистернах
- vagonbrak_prt - браки
-vagonplomb - пломбы
- vagonsl - данные сортировочного листка
-vagonoper - времена учетных операций с вагоном на станции
- vagon_otpr - основные данные об отправке

Технологический уникальный ключ- инвентарный номер вагона. В модели может быть только один вагон с данным инвентарным номером. Есть система решения конфликтов, если один вагон одновременно числится на разных станциях.

При вставке вагона ему автоматически присваивается системный идентификатор VagId. Через него идет склейка дочерних таблиц.

В дорожной модели вагоны удаляются по отправлению с дороги и вновь вставляются по приходу. Вставляются с новым ИД.

В сетевой модели вагоны никогда не удаляются, даже если отправлены за границу.

Вагон не имеет собственной дислокации. Взамен этого он постоянно имеет ссылку на «хозяина», у которого есть дислокация. Ссылка на хозяина это: тип EntitType и идентификатор EntitId.

Перечень типов:

- 1 – поезд
- 2 – место подачи
- 4 – дальнее информирование(2977), для дорожных БД
- 5 – оформлена отправка, других сведений пока нет
- 8 – отправлен за границу
- 9 – удален

- **Описание вагонного архива**

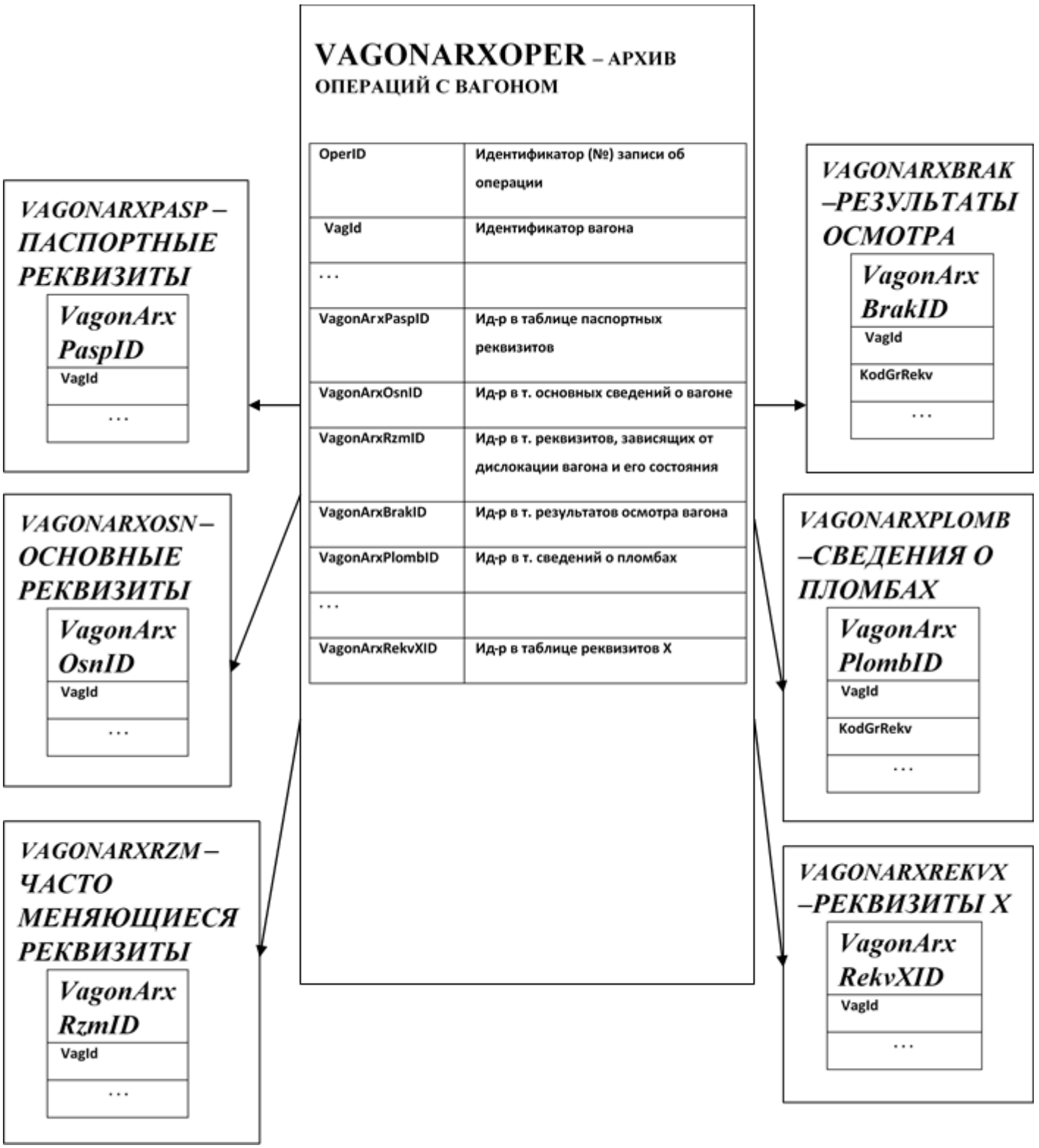


Рисунок 13

На рисунке 13 приведено схематическое описание вагонного архива.

Центральная таблица VagonArxOper – архив операций с вагоном. Ключом таблицы является идентификатор или номер записи об операции с вагоном. Таблица содержит непосредственно сведения об операции (код, время, автор), текущий и предыдущий объекты принадлежности вагона и идентификаторы связи или ссылки на реквизиты, расположенные в других таблицах архива. Введено также поле OperOtmID – идентификатор записи об операции, отменяющей данную операцию, которое необходимо для «путешествия во времени».

На сегодняшний день выделено 5 таблиц реквизитов. Основной принцип размещения полей по новым таблицам – частота их изменения. В отдельные таблицы могут быть вынесены автономно меняющиеся группы реквизитов, которые, независимо от частоты их изменения, изменяясь сами, не затрагивают других реквизитов.

Таблица паспортных реквизитов VagonArxPasp будет содержать практически неизменные сведения о вагоне из картотеки АСОУП и таблиц технических характеристик НСИ АСУ ЖДП. Чаще всего за все время существования вагона в системе ОЦ здесь будет лишь одна строка сведений.

Таблица основных реквизитов VagonArxOsn является самой объемной таблицей и включает сведения о вагоне, отражающие состояние вагона относительно погруженного груза, возможность использования его для последующей погрузки.

В таблице с часто изменяющимися реквизитами VagonArxRzm предполагается сохранять данные о вагоне, которые зависят не только от состояния вагона, но и от его дислокации. Здесь будут постоянно пересчитываться поля разметки, а также признак необходимости охраны.

Следующие 2 таблицы о результатах осмотра VagonArxBrak и пломбах VagonArxPlomb имеют структурное отличие от выше описанных таблиц, поскольку содержат несколько строк по вагону (с одним VagID) для одного и того же идентификатора связи (VagonArxBrakID, VagonArxPlombID). Поэтому в них вводится еще одно ключевое поле Идентификатор Группы Реквизитов.

- **Описание поездной модели**

Модель состоит из главной таблицы kcmod.train и набора дочерних:

- trainsoed - соединенные работа
- trainoper - времена учетных операций с поездом на станции

Технологический уникальный ключ- 6-значный индекс поезда. В модели может быть только один поезд с данным индексом.

При вставке поезда ему автоматически присваивается системный идентификатор TrnId. Через него идет склейка дочерних таблиц.

Группа вагонов на станционном пути также считается поездом, ей соответствует своя запись о поезде с условным индексом.

Алгоритм работы поездного архива такой же, как и вагонного. Таблицы - **TrainArxOper, TrainArxOsn, TrainArxRazl**

3.4.2 СЕРВЕР ПРИЛОЖЕНИЙ

Сервер Приложений, отвечающий за работу с клиентскими местами, осуществляет хостирование приложений: поддержку их в работающем состоянии, масштабирование и т.д. Принимает запросы от клиентских мест, переадресует их приложениям. Организует взаимодействие между приложениями, планировщиком и БД.

В АСУ ЖДП могут выполняться десятки различных операций. Под операциями понимаются не только прибытия и погрузки, но и ввод НСИ, взятие справки

Все эти операции разбиты на функциональные блоки: ДСП, ТК, НСИ, Справка и т.д.

Блоки операций определяют соответствующие роли. Роли используются для назначения прав пользователям.

Одна операция может входить в несколько блоков, например подача-уборка должна входить и в ТК и в Приемосдатчик.

Каждый блок операций реализуется с помощью своего типа WEB-АРМ: WEB-ДСП, WEB-ТК, WEB-Справка…

У каждого типа WEB-АРМ есть своя стартовая страница, разработанная под удобство выполнения задачи этого АРМ.

У каждого WEB-АРМ есть свое Лог.Имя в ЦТО. Наличие собственного Лог.Имени позволяет подписаться на получение квитанций на обновление только для активного в это время WEB-АРМ. Свое Лог.Имя – это свой автоответ, свои права на передачу сообщений в АСОУП.

Внутри АСУ ЖДП WEB пользователь, согласно полномочий юзера, имеет доступ к различным Веб-АРМам.

Работа пользователя начинается с авторизации в АСУ ЖДП WEB.

Далее на экране появляется каркас АРМ, стартовая страница WEB-АРМ и список разрешенных к работе других типов WEB-АРМ. Какая из страниц WEB-АРМ стартовая – настраивается.

Операции можно выполнить двумя путями:

- перейти на страницу соответствующего WEB-АРМ

- вызвать напрямую операцию через меню текущей страницы

При загрузке АРМ в адрес планировщика идет сообщение о регистрации юзера и WEB-АРМ.

При переходе на страницу другого WEB-АРМ, новый WEB-АРМ регистрируется, предыдущий отключается.

Аналогичные операции происходят относительно юзера, в результате известно с какими WEB-АРМ, в какое время работает или работал конкретный пользователь.

WEB-АРМ в данный момент может быть зарегистрирован за одну станциями. При выборе новой станции посылается сообщение Планировщику о перерегистрации. Это относится к поездным АРМ.

АСУ ЖДП WEB читает информацию из базы данных PG SQL. Для записи в базу АСУ ЖДП WEB посылает сообщение Планировщику. При необходимости Планировщик посылает сообщение в АСОУП от имени Веб АРМ. Веб-АРМ напрямую не посылает сообщение оперативной работы в АСОУП, но может напрямую запрашивать справки из АСОУП.

3.4.2.1 Термины

Юзер – понятие используемое при авторизации (логин/пароль). Юзер имеет свои полномочия (Роли, операции, инфраобъекты).

Лог-имя (далее LogName) – идентификатор абонента в системе телеобработки АСУ ЖДП.

Тип АРМа– определяется технологическим типом АРМ и его программной реализацией (WinTK, WebTK). В таблице citns.ArmType это поле ArmType_id. В таблице Citnv..ArmUserNet это TypeArm. У всех веб-АРМов тип лежит в диапазоне 200-255

АСУ ЖДП WEB – комплекс программ, устанавливаемых на сервере, обеспечивающий функционал работы с тонким клиентом. АСУ ЖДП WEB работает под управлением WEB-сервера Nginx. Состоит из оболочки и компонентов.

Компонент – комплекс программ в составе АСУ ЖДП WEB обеспечивающий функционал работы одного АРМа в старом понимании (АРМ ТК, АРМ ТОВ, Справочная система и т.д.)

Оболочка – комплекс программ в составе АСУ ЖДП WEB обеспечивающий координацию, и общие функции Веб-АРМов, такие как авторизация, загрузка НСИ, печать, логгирование, сохранение пользовательских настроек, стиль элементов управления и др.

Страница – видимый в окне браузера интерфейс, обеспечивающий взаимодействие с пользователем (клиентский комп), и выполняющий одну технологическую задачу.

Сайт - комплекс всех страниц, предоставляемых АСУ ЖДП WEB. Состоит из веб-АРМов и страниц оболочки.

Каркас – совокупность общих для всех страниц сайта элементов, таких как заголовок, меню выбора страниц, меню действий пользователя, строка состояния, нижняя панель реквизитов (footer) и т.д. Формируется оболочкой

Контент - функциональное содержимое страницы. Формируется компонентом. Страница обычно состоит из каркаса и вложенного в него контента.

Веб-АРМ – комплекс страниц, обеспечивающий функционал работы одного АРМа в старом понимании (АРМ ТК, АРМ ТОВ, Справочная система и т.д.)

3.4.2.2 Обмен сообщениями

Подключение.

Пользователь набирает в браузере адрес сайта https://172.17.131.170/cittrans/asust и попадает на страницу авторизации.

При первой авторизации со своим юзером он попадает на стартовую страницу сайта АСУ ЖДП WEB, на которой ему предлагается перейти на

страницу настройки и там настроить порядок, видимость и стартовую страницу доступных ему, согласно полномочий юзера, веб-АРМов. В дальнейшем, после авторизации пользователь сразу переадресуется на настроенную стартовую страницу веб-АРМа.

Прямое сообщение (Страница > Планировщик).

При первой загрузке веб-АРМа со своим типом TypeArm1 в АСУ ЖДП WEB в адрес планировщика формируется сообщение 4151 на регистрацию в таблице ArmWork. По паре параметров: предоставленному веб-АРМом *muny*, и определённому в АСУ ЖДП WEB *HostName* из таблицы Citnv..ArmUserNet выбирается лог-имя (LogName) веб-АРМа. Также в АСУ ЖДП WEB определяется необходимый для сообщения 4151 идентификатор юзера.

Когда пользователь инициирует сообщения из веб-АРМа (на запись, на получение справки из АСОУП и т.д.) оно формируется в АСУ ЖДП WEB согласно определённому ранее *LogName* и через ЦТО посылается адресату (Планировщик, АСУ ЖДП и др.)

Обратное сообщение (Планировщик (АСОУП) > Страница)

Обратное сообщение (квитанцию, сообщение на обновление) Планировщик и АСОУП посылает на имя LogName веб-АРМа, но, поскольку веб-АРМы и АСУ ЖДП WEB собраны в одну группу ТО (см. п. 2.2), по факту оно приходит в АСУ ЖДП WEB.

По приходу сообщения АСУ ЖДП WEB, используя связку *mun-HostName-LogName* (см. п. 2.3), инициирует событие получения сообщения в веб-АРМе. Где оно обрабатывается соответствующим образом.

3.4.3 ПЛАНИРОВЩИК

3.4.3.1 Назначение

Планировщик работает в составе АСУ ЖДП. Планировщик состоит собственно из планировщика и обработчиков сообщений. Планировщик

получает через телеобработку сообщения и передает их на обработку соответствующему обработчику, отслеживая работоспособность обработчиков. Планировщик с обработчиками сообщений практически владеют монопольным правом на корректировку данных в БД АСУ ЖДП.

3.4.3.2 Функциональные возможности

Реализован запуск таймерных задач, написанных на языках T-SQL и C#. Деление обработки сообщения на два этапа: предварительный и основной.

Возможность одновременного функционирования нескольких экземпляров планировщика, каждый со своей очередью сообщений и со своим логименем.

Планировщик следит за временем обработки сообщений (можно для каждого сообщения указать свое время) и, если нужно, перегружает долго работающий или “упавший” компонент.

Реализована возможность параллельной обработки нескольких сообщений.

3.4.3.3 Описание программы

Программа предназначена для распределения входящих сообщений между подпрограммами обработки, выполненных в виде библиотек. Библиотека может включать в себя один или несколько обработчиков сообщений. Подпрограмма может быть написана на языках C++, C#, Java. Для каждого языка был написан компонент, который и загружает библиотеки.

Каждый компонент загружен в своем адресном пространстве, что исключает возможность порчи данных других компонент.

Обмен информацией между планировщиком и компонентами реализован по TCP/IP. Компонента предоставляет некоторые сервисные

функции типа: посылки сообщений абонентам, рассылка сообщений АРМ, работающих на конкретной станции, доступ к таблицам НСИ, загруженным в память и т.д.

Каждая компонента может быть загружена несколько раз и ей может быть назначен свой набор сообщений. Это дает большую гибкость при исправлении ошибок (не надо менять все сразу, достаточно будет заменить конкретный обработчик), подразумевает наличие одной телеобработки (кроме центральной) для всех приложений, включенных в состав планировщика.

Программа выполнена в виде полноценной службы и имеет в своем составе службу и консоль управления службой.

3.4.3.4 Порядок запуска планировщика

Для запуска планировщика необходимо настроить таблицы НСИ nvSrvComp, nvIP и три таблицы: nsSrvComp, nsSrvMsg, nsSrvMsgXML должны быть настроены разработчиками в ЦИТТРАНС

3.4.3.5 Настройка НСИ

Таблица nsSrvComp.

Таблица описывает все возможные компоненты.

progId - программный идентификатор com-класса компонента

compId - идентификатор компонента (должен быть больше 0)

name - название компонента

compType - тип объекта(0-обычный,1-обрабатывает сообщения в отдельном потоке,2-не принимает сообщения)

isMain - не принимать сообщения , если этоткомпонент не смог загрузиться

Пример:

progId	compId	name	compType	isMain
Server.pid	1	Сервер приложений	0	1
Accounts.pid	2	Отчетность	0	1
Spravka.pid	3	Справочная система	0	1

Таблица nsSrvMsg.

Сопоставляет код сообщения и компонент, который обрабатывает это сообщение.

msgNo - код сообщения. Должен указываться с ведущими нулями. В любой позиции номера сообщения может стоять символ '?', означающий любой знак.

compId - идентификатор компонента (из таблицы nsSrvComp)

orderNo - порядковый номер обработки сообщения.

maxTime - максимально допустимое время (в сек) обработки сообщения.

Таблица настраивается разработчиками.

Пример:

msgNo	compId	orderNo	maxTime
????	1	1	120
0777	3	1	240

Таблица nvSrvComp.

Описывает компоненты, загружаемые на данном объекте автоматизации.

compId - идентификатор компонента (из таблицы nsSrvComp)

name - название компонента(если при заполнении не введено, заносится из таблицы nsSrvComp)

Таблица настраивается для конкретной станции.

Пример:

compId	name
1	Сервер приложений
3	Справочная система

Таблица nvSrvMsg.

Дает возможность изменить время обработки конкретного сообщения.

msgNo - код сообщения. Должен присутствовать в nsSrvMsg.

compId - идентификатор компонента (из таблицы nsSrvComp)

maxTime - максимально допустимое время (в сек) обработки сообщения.

Таблица nvSrvMsgDop.

Дает возможность изменить время обработки сообщения 504 и 1062 по коду операции.

msgNo - код сообщения. Должен присутствовать в nsSrvMsg.

msgNoUt - код операции в сообщении

maxTime - максимально допустимое время (в сек) обработки сообщения.

Таблица nsSrvMsgXML.

Сопоставляет код и xml сообщения по вхождению прописанной строки.

msgNo - код сообщения. Должен указываться с ведущими нулями.

messPart – строка

Таблица nvIP.

Необходимо прописать IP адрес SQL сервера, с которым работает ЦТО (где хранятся очереди ЦТО), описание заполнения смотри в «Таблица NvIP, описание серверов и баз АСУ ЖДП (бывшая nvSPOR).doc»

3.4.4 ЦТО (ЦЕНТРАЛЬНАЯ ТЕЛЕОБРАБОТКА)

Центральная Телеобработка отвечает за гарантированную доставку сообщений:

- внутри системы
- внешним системам, по различным протоколам
- при обмене с действующей АСУ ЖДП

Наличие сообщений в системе обеспечивает надежность (гарантированная доставка) и гибкость (например взаимодействие с действующей АСУ ЖДП и другими системами).

НАЗНАЧЕНИЕ: работа в качестве центральной телеобработки как под Windows так и под Linux

ЯЗЫК ПРОГРАММИРОВАНИЯ: C# NET CORE 3.1 (хотя, возможно, достаточно меньшей версии)

Реализована в виде библиотеки классов. Стартовый класс запускается сторонним приложением (на момент - сделано консольное приложение для запуска ЦТО).

3.4.4.1 Работа ЦТО

При старте, ЦТО загружает параметры работы (место хранения очередей, логов, имя таблицы с абонентами, список адресов "админов", число

дней хранения логов и сообщений, списки кодов сообщений в/из АСОУП для пересылки в СП) из файла настроек.

Очереди абонентов хранятся в виде файлов. По 2 файла на каждый день хранения для каждого абонента. Один файл - массив структур описателей сообщений, второй - собственно сообщения. Файл создаётся размером с запасом на день. Размер вычисляется на основе статистики по предыдущим дням. По окончании суток файлы обрезаются под фактически занятое место. Пустая очередь или с истёкшим лимитом времени хранения удаляется.

Сообщение имеет следующие атрибуты:

string Autor[50]; отправитель

string Abonent[50]; получатель

string Code[6]; код сообщения

string Prefix[100]; префикс

byte Atribute, атрибут

byte RecordType; состояние обработки

UInt32 Len; длина

byte CodePage;//Кодовая страница сообщения

long TimeReceive; время прихода

long TimeUpdate; время окончания обработки

UInt32 TimeProcessing; время обработки

Guid SenderID, ReplyID; уникальный Guid отправителя и ссылка на сообщение-инициатор.

UInt64 MesID - номер сообщения в очереди

Атрибуты Autor Abonent Prefix хранятся в UTF-8, поэтому, в байтах под них зарезервировано в 2 раза больше указанного количества. Предельное число символов получается зависящим от самих символов: чисто латиницы может быть в 2 раза больше.

ТО СП умеет от рождения, а NetTo.DLL доработана для поддержки работы с SenderID, ReplyID и расширенным Code;

На момент, ЦТО поддерживает 2 типа абонентов: STDP клиент и STDP сервер. Сама, соответственно, может выступать в качестве STDP клиента и STDP сервера. Список абонентов загружается из PG из таблицы citnv.armusernet. Для удобства ведения списка абонентов, абоненты с номером группы "0" - считаются абонентами с индивидуальными подключениями так же как и абоненты с уникальным номером группы. Абоненты с одинаковой группой (но не 0) - считаются находящимися в одной группе и очередь ведётся на 1-го абонента в группе.

Для поддержки создания кеша сообщений на приём в ТО СП, ЦТО поддерживает опережающую отправку сообщений, не дожидаясь обработки предыдущих.

Просмотр состояния каналов, очередей, поиск сообщений и т.п. осуществляется управляющими командами, передаваемыми через TCP сервер, выполняющий роль единой точки подключения как STDP клиентов так и Консоли. Чистка очередей разрешена только администраторам, адреса которых внесены в настроечный файл (если указаны, иначе - всем).

3.4.4.2 Установка ЦТО

Нижеописанные действия необходимо выполнять под тем пользователем под которым будет запускаться ЦТО либо давать ему необходимые права на каталоги или файлы.

1. Создать каталог для программы и скопировать в него содержимое "\CIT-NEW\Свободное ПО\DISTRIB\CTO". Изменить установленный фреймворк на NET CORE 3.1 в файлах *.json, если вместо NET5 установлен NET CORE 3.1

2. Отредактировать файл settings.ini в части указания местоположения каталогов для очередей, логов и строки подключения к базе. Пример настроечного файла(кодировка UTF-8):

{
"КаталогЛогов":"E:\\MyFiles.C\\CTO_CORE\\logs",
"КаталогОчередей":"E:\\MyFiles.C\\CTO_CORE\\Queue",
"СписокАдресовАдминов":"10.200.66.162,172.17.131.117",
"History":"8",
"СтрокаПодключения":"Host=172.17.131.171;Port=5432;Username=postgres;Password=CITdl160adm;Database=asust;",
"КодыКопийСообщенийВАСОУП":[208,449,1042,02,09,209,230,333,12,1353,1354,1397],
"КодыКопийСообщенийИЗАСОУП":[208,449,1042,02,09,209,230,333,12,1353,1354,1397]
}

Минимально необходимое для запуска - СтрокаПодключения. Пользователь из этой строки должен иметь права на чтение таблиц ArmUserNet, Citnv.nvsendsoob, Citnv.nvip, citnv.nvservernastr, citnv.srvlogimps, kcmod.mtomessagequeue, kcmod.mtomessagequeue. Последние 2 ещё и на изменение.

3. Создать (изменить, если есть) в таблице абонентов(armusernet) строку(и) с STDP сервером(ами) к которым будут подключаться абоненты. Для STDP серверов важны параметры: Автоответ (13 символов, включая 2 пробела), используемый адрес (конкретный или 0), порт, признак включенности (enabled). linetype=10, typearm=202. Пример:

SELECT * FROM citnv.armusernet WHERE linetype=10

idarm logname autoanswer linetype codepage signpersonaltelesoob userdirin signrestorequeue signstoresoob addres port description enabled groupingid typearm

56 stdptest (?? ???? ?)01 10 1 1 0 0 0 0 8886 ЦТО_PG: Сервер STDP 1 0 202

4. Прописать АРМЫ - клиенты STDP. Для STDP клиентов важны параметры: Логимя, Автоответ (13 символов, включая 2 пробела), кодировка, период проверки канала в сек. signpersonaltelesoob, контроллируемый адрес(конкретный или *), группа STDP(0 или конкретная), признак включенности (enabled). linetype=11 или 14(обычно 14), typearm в зависимости от типа АРМа. Пример:

SELECT * FROM citnv.armusernet WHERE linetype in (11,14)

idarm logname autoanswer linetype codepage signpersonaltelesoob userdirin signrestorequeue signstoresoob addres port description enabled groupingid typearm abonentvid esr nomzayvki mnkd

42 ps_osnudk (PS ???? U)DK 14 3 1 180 0 0 * 0 Планировщик Устинов 1 NULL 201 0 NULL NULL NULL

37 jok2_astra (?? ???? ?)37 14 3 1 180 0 0 * 0 Планировщик Голышев 1 0 201 0 0 NULL Jok

14 CHUB_TK_CHUB (?? ???? ?)14 14 3 1 180 0 0 CHUB-HP 1 АРМ ТК у Чубчева локальный 1 0 225 0 0 ТКЧуб

8 CHUB_DSP (?? ???? ?)07 14 3 1 180 0 0 CHUB-HP 1 АРМ ДСП у Чубчева локальный 1 0 204 0 0 ДСПЧуб

4 CHUB_RS_CHUB (?? ???? ?)04 14 3 1 180 0 0 CHUB-HP 1 АРМ РС у Чубчева локальный 1 0 205 0 0 РСЧубЛок

1 ASUSTWEB_CHUB (?? ???? ?)03 14 1 1 180 0 0 172.17.131.142 1 АСУ ЖДП ВЕБ на машине у Чубчева 1 0 200 0 0 АСУ ЖДПВЕБЧуб

5. Если необходимо прописать подключение к другому STDP серверу, нужно, как и в старой ЦТО, либо прописать канал связи(linetype=0) и абонентов за ним(linetype=1) либо абнентов-STDP серверов (linetype=21 или 22(обычно 22)). В первом случае очередь ведётся на канал. Во втором случае абоненты с совпадающими адресами и портами будут считаться в одном канале. Подключение и ведение очереди - по первому абоненту в канале. В

качестве собственного автоответа, при подключении, используется логимя. Поэтому, оно должно удовлетворять требованиям к автоответу. Пример:

SELECT * FROM citnv.armusernet WHERE linetype in (0,1,21,22) order by 1

idarm logname autoanswer linetype codepage signpersonaltelesoob userdirin signrestorequeue signstoresoob addres port description enabled groupingid typearm abonentvid esr nomzayvki mnkd

7 СТО ASTRA 000 (8 00000 8)00 0 3 0 0 0 0 172.17.170.23 8885 ЦТО Луга 1 NULL 202 0 NULL NULL NULL

26 208 (?? ???? ?)?? 1 3 1 1 0 0 * 1 ASOUP 1 0 11 0 0 NULL NULL

39 (? STAT ASTRA (?? ???? ?)CI 22 3 1 -5 0 0 10.200.66.120 8885 Статистика 1 NULL 206 0 NULL NULL NULL

40 CITTRANS (?? ???? ?)C0 22 3 1 -5 0 0 10.200.66.120 8885 Статистика 1 NULL 206 0 NULL NULL NULL

Для запуска ЦТО необходимо создать на рабочем столе иконку с параметрами запуска. Пример:

"dotnet /home/administrator/cittrans/CentrTO/ConsoleApp1.dll"

3.4.5 WEB АРМ (ПРИЛОЖЕНИЕ АСУ ЖДП WEB)

Основное приложение системы - АСУ ЖДП WEB. Состоит из стандартного каркаса(оболочки) и набора прикладных страниц. Любой АРМ строится на основе общего каркаса.

Реализована возможность WEBAPM, в рамках взаимодействия с ЦТО, иметь свой автоответ и логическое имя. Это позволяет WEB АРМ регистрироваться в системе, отправлять сообщения, получать квитанции и уведомления. Эти функции реализованы в рамках приложения ASUSTWEB.

Каркас реализует общие для всех АРМ функции:

- Авторизация
- ЭП
- Обмен сообщениями с планировщиком
- Обеспечение прикладных страниц таблицами стилей и популярными скриптами
- Кэшированная НСИ
- Обработка ошибок
- Логгирование
- Сохранение настроек
- Печать

3.4.6 ПОДСИСТЕМА ЗАПУСКА ТАЙМЕРНЫХ ЗАДАЧ(JOB)

В связи с тем, что в Postgres нет механизма запуска таймерных задач, в планировщике создан механизм, который может запускать по расписанию SQL команды. Для запуска таких(SQL) команд, необходимо прописать их в таблицах описания таймеров, выставив признак iscommand = 1.

После описания нового таймера, необходимо перезагрузить планировщик, для обновления библиотек.

3.4.6.1 Описание НСИ и принципов работы таймеров.

Создана таблица НСИ, в которой должны быть описаны все JOB:

```
CREATE TABLE citns.srvtimers (
  timerid int4 NOT NULL,--идентификатор таймера
  timertype int2 NOT NULL DEFAULT 0,--тип таймера
  timeday int4 NOT NULL DEFAULT 0, --в зависимости от типа
  timehour int2 NOT NULL, --в зависимости от типа
  timeminute int2 NOT NULL, --в зависимости от типа
```

maxtime **int4 NOT NULL DEFAULT** 0,--максимальное время работы(в чем)

description **varchar NULL**,--описание

iscommand **int2 NOT NULL** , 1 -выполняется заданная SQL команда.

commandtext **text NULL**, --выполняемая команда

CONSTRAINT pk_nssrvtimers **PRIMARY KEY** (timerid)

);

TimerId - для чисток от 100 000, для других – свой уникальный

тип таймера:

0 - Интервальный,

1 – По дням недели,

2 - Месячный

Интервальный – Частота выполнения таймера задается полями timehour, timeminute.

По дням неделий – Дни недели запуска задаются битами в поле timeday. Время запуска в полях timehour, timeminute.

Месячный - Дни месяца запуска задаются битами в поле timeday. Время запуска в полях timehour, timeminute.

Задание само отвечает, за то, чтобы отрабатывать только на нужном типе сервера (АСУ ЖДП, ССП, ТК….)

Все задания, работу которых надо временно отключить или увеличить лимит времени, должны быть описаны в таблице citnv.srvtimers:

CREATE TABLE citnv.srvtimers (

timerid **int4 NOT NULL**,

esvkl **bool NOT NULL DEFAULT** 0::**boolean**,-- 0- отключен, 1 - включен

maxtime **int4 NULL**,

CONSTRAINT pk_nvsrvtimers **PRIMARY KEY** (timerid)

);

3.4.6.2 Наблюдение за работой заданий.

Предлагается сделать задания полноценным участником технологического процесса АСУ ЖДП. Т.е. предлагается каждому заданию иметь свой технологический код, по аналогии с кодом сообщения. Причем коды сообщений и коды заданий должны иметь сквозную нумерацию (диапазон 7000-7500).

Для ведения модели создаем таблицу Kcmod.StatTimers.

CREATE TABLE kcmod.stattimers (

timerid **int4 NOT NULL**, - ИД таймера

laststarttime **timestamp NULL**, - Время последнего запуска таймера

lastendtime **timestamp NULL**, - Время последнего окончания таймера

isfault **int2 NULL**, - статус (0 – успешно завершен, 1- завершен с ошибкой, 2- работает) статус меняется раз в минуту.

lastresult **varchar NULL**- текст ошибки

);

CREATE UNIQUE INDEX ix_stattimers_timerid **ON** kcmod.stattimers **USING** btree (timerid);

Для ведения истории выполнения заданий создается таблица Kcmod.StatTimersArx.

CREATE TABLE kcmod.stattimersarx (

timerid **int4 NOT NULL**, - ИД таймера

starttime **timestamp NULL**, - Время запуска таймера

worktime **int4 NULL**, - Продолжительность работы

isfault **int2 NULL**, - признак завершения (0- успешно, 1- ошибка)

lastresult **varchar NULL**- Текст ошибки

);

CREATE INDEX ix_stattimersarx_timerid **ON** kcmod.stattimersarx **USING** btree (timerid);

Для выявления одновременно работающих заданий, используем просмотр kcmod.vwstattimersarx. Он показывает не все варианты одновременной работы, но для оценки более чем достаточно. Для текущего состояния kcmod.vwstattimers.

Начало и конец работы таймера также фиксируется в таблице kcmod.journal, с кодами 4 и 5.

Появилась возможность менять время запуска и (только для таймеров, выполняющих команды SQL) текст команды на ходу, без перезагрузки ПС. Включение- отключение таймеров без перезагрузки не возможно.

3.4.7 АВТОРИЗАЦИЯ В БД АСУ ЖДП PG.

В настоящее время в БД АСУ ЖДП PG существует одна учетная запись «postgres», с правами суперпользователя, т.е. с доступом везде и на все, включая право на удаление БД.

В АСУ ЖДП разработана система доступа(авторизации) к БД, с набором пользователей, с набором ролей, позволяющую разделять полномочия между **разработчиками** АСУ ЖДП, а также между **подсистемами** АСУ ЖДП.

В основе системы авторизации будет положено разделение доступа к схемам.

Предложения:

1. Создаем набор учетных записей (УЗ) для разработчиков с правом доступа к одной или нескольким схемам, связанных с одной подсистемой АСУ ЖДП:

- грузовая модель
- поездная и вагонная модель
- контейнера
- ТОВ
- т.д.

2. В основе деления полномочий лежит следующее:

- роль имеет **все** полномочия относительно объектов внутри своих схем
- может читать данные из таблиц чужих схем
- может выполнять процедуры схемы КСМОД (запись в журнал, корректировка вагона и т.д.)

3. При необходимости можно делить полномочия ролей до уровня отдельных таблиц и процедур.

4. Создаем набор учетных записей (УЗ) для приложений.

- приложение «планировщик системы»
- приложение «АСУ ЖДП WEB»
- приложение «ЦТО»
- и т.д.

5. Создание такого набора пользователей потребует некоторого времени и может вызвать на первом этапе определенные неудобства разработчиков. Надо понимать, что разделение полномочий невыгодно практически каждому отдельному разработчику, но это выгодно системе в целом.

6. По окончании переходного периода у пользователя «postrres» меняется пароль. И его будут знать только админы системы: Бережной, Максимов, Томин.

Реализация:

Создается две группы УЗ. Для разработчиков, начинаются на u (user) и для приложений, начинаются на a (application).

УЗ Для разработчиков:

ukcmod -- поездная работа

- Полный доступ к схемам:kcmod,citns,citnv,fdw,tools
- Чтение других схем.

ucargo -- грузовая работа

- Полный доступ к схемам gkmod,tax,rastcit,tools,fdwgk ,orgcl
- Чтение: kcmod,citns,citnv,fdw
- Полный доступ к технологическому НСИ

ukont; -- контейнерная работа

- Полный доступ к схемам kont,kontarx,kontbad,kontold,tools
- Чтение: kcmod,citns,citnv,fdw
- Полный доступ к технологическому НСИ

Для приложений созданы свои учетные записи:

aps - планировщик

acto - ЦТО

aweb - приложение АСУ ЖДП WEB

Предполагается, что создание новых таблиц в схемах citns, citnv, fdw будет происходить через подачу заявок. Все неописанные объекты переместятся в схему tmp.

3.4.8 МЕНЕДЖЕР ПАРОЛЕЙ АСУ ЖДП ДЛЯ ПОДКЛЮЧЕНИЯ К БД.

Решаемая Задача:

1. Создание набора учетных записей(УЗ), для всех типов приложений АСУ ЖДП, с учетом необходимых прав приложений на работу с объектами БД.
2. Отказ от существующего набора УЗ (spks,ks,ksacc, …), основанного на фиксированных паролях УЗ.
3. Обеспечить возможность ввода пароля для учетных записей, в соответствии с требованиями информационной безопасности.
4. Возможность использования разного набора паролей учетных записей на разных узлах системы
5. Обеспечение возможности периодической смены пароля учетных записей MS SQL, используемых для подключения приложения АСУ ЖДП к БД PG.
6. Использовать для шифрования паролей только сертифицированные средства.
7. Обеспечить работу АСУ ЖДП на время переходного периода, в течении которого приложения в плановом порядке переводится на работу с новой системой авторизации.
8. Обеспечить работу АСУ ЖДП в аварийном режиме, если система генерации и чтения паролей начала сбоить.
9. Реализовать автоматическую доставку файла с новым паролем до приложений и переход на работу с новым паролем.

Реализация:

1. Для каждого типа приложения АСУ ЖДП, серверного и клиентского, выделяется отдельная учетная запись(пользователь) PG. Эти пользователи и доступные им роли создаются при генерации системы.

2. Связь типа приложения и его учетной записи в PG прописана в классификаторе citns.armtype.

3. Разрабатывается отдельное приложение для корректировки паролей. Это приложение также генерирует config-файлы, отдельно для каждого типа приложений. Этот файл будет содержать в зашифрованном виде логин и пароль для подключения к БД. Считаем, что серверные приложения работают на разных серверах. И админу необходимо вручную обновлять config-файлы на серверах.

4. Серверные приложения хранят пользователя и пароль в настроечных файлах в зашифрованном виде. В Linux есть набор фиксированных каталогов, поэтому можно выбрать одно место для расположения этого файла. Для шифрования выбираем стандартный метод, доступный и для C++ и для C# или пишется библиотека на C++. Ключом для шифрования предлагается выбрать файл citini.cit – лицензию ЦИТТранс.

5. На серверах создается каталог с фиксированным именем, в котором будут хранится config-файлы с зашифрованным логин и пароль для каждого серверного приложения. Такой каталог необходим, т.к. серверные приложения работают как службы, стартуют автоматически и пароль вводить некому.

6. Практически все АРМ реализованы по технологии WEB-АРМ. Поэтому прямое подключение к БД им не требуется, они работают через приложение АСУ ЖДП-WEB.

7. Если будут «толстые клиенты», то сейчас все рабочие места обязаны проходить авторизацию РЖД, с выдачей прав на выполнение набора операций. Т.Е. они вызывают службу авторизации АСУ ЖДП, передавая на

вход логин и пароль АСУ ЖДП, а также тип приложения из таблицы citns.armtype. Служба в результате выдает пользователя и пароль на подключение к базе данных.

8. В случае поломки менеджера паролей, админ может вручную поменять пароли для серверных приложений и записать их в конфиг-файлы приложений. Это временный режим, на время устранения неисправности. См инструкцию: \CIT- NEW\Свободное ПО\АСУ ЖДП 1 этап работ\Менеджер паролей АСУ ЖДП

9. По команде «Сменить пароль приложения» в адрес приложения высылается сообщение с файлом пароля и меняется пароль в БД. Приложение, получив сообщение, автоматом закрывает старые соединения и открывает новые.

10. Серверные приложения, для дополнительной защиты, могут хранить файлы с паролями, в директориях доступных только пользователю приложения.

3.4.9 РЕГИСТРАЦИЯ АРМ И ПОЛЬЗОВАТЕЛЕЙ

Решаемая задача:

- В АСУ ЖДП существует набор АРМ. Они описаны в таблице Citnv.ArmUserNet.

- В АСУ ЖДП также есть набор пользователей. Описаны в таблице Citnv.User.

- Один пользователь может работать одновременно на нескольких АРМ.

- Есть загрузка и выгрузка АРМ, которая автоматом вызывает регистрацию/разрегистрацию пользователя.

- АРМ может быть зарегистрирован за одну станцию, необходима возможность ее смены.

- Есть смена пользователя без перезагрузки АРМ.

- Для рассылки квитанций необходима модель загруженных АРМ.

- Необходима отдельная модель пользователей, т.к. события с объектом «пользователь» не обязательно связаны с АРМ. Таблица работников nvLpr удалена. В таблице Citnv.User описаны не только работники АСУ ЖДП. Значит пользователи могут выполнять и другие операции: подписывать документы, ставить башмаки и т.д. Возможно потребуется отслеживать дислокацию работников, без модели пользователей это невозможно.

Реализация:

1. Для ведения модели АРМ создается таблица kcmod.ArmWork, структура:

IdArm	ИД АРМ (из CitNv..ArmUserNet
UserId	ИД пользователя (из CitNv..Users)
TypeArm	ТипАРМ (Из CitNs.ArmType)
Esr	ЕСР
Oper	Код операции:

1 - Загрузка АРМ

3 - Смена дислокации

5- Перезагрузка

DateSys	Системная дата операции.

Архив для ArmWork не планируется. По выгрузке АРМ строка удаляется.

1. Для ведения модели пользователей создается таблица kcmod.UserWork, структура:

UserId	ИД пользователя (из CitNv..Users)

Esr	ECP
Oper	Код операции:

6- Начало работы пользователя

7- Завершение работы пользователя

DateSys	Системная дата операции

Началом работы пользователя, код = 6, считается первая загрузка АРМ от его имени.

Окончанием работы пользователя, код = 7, считается последняя выгрузка АРМ от его имени.

Архив UserWorkArx , его структура:

UserId	ИД пользователя (из CitNv..Users)
IdArm	ИД АРМ (из CitNv..ArmUserNet
Esr	ECP
Oper	Код операции;
DateSys	Системная дата операции

Если пользователь, в течении смены работал на разных АРМ, то IdArm начала и конца работы пользователя может не совпадать. Это некритично, т.к. не ставится задача отследить точное время работы пользователя на каждом АРМ.

2. Для регистрации/разрегистрации АРМ и пользователей используется сообщение 4151. Структура сообщения:

Структуру сообщения менять не стал, не критично, а АРМы уже разошлись по рабочим местам

(:4151 UserID ArmID datebegin - objid Oper CHUB-HP :)

UserID - идентификатор пользователя

ArmID – идентификатор АРМа

Datebegin - 2023-02-07_17:25:55 дата загрузки

«-» - не используется

Objid – id станции регистрации АРМ

Oper – операция, где:

1 - Загрузка АРМ

0 - Выгрузка АРМ.

2 - Смена дислокации

3 - Смена пользователя на данном АРМ

3. Обработка с.4151:

- Если *загрузка выгруженного АРМ*, то ArmWork.Oper = 1

Если загрузка первого АРМ, где работает данный пользователь, то UserWork.Oper = 6.

Иначе: перезагрузка АРМ ArmWork.Oper = 5.

- Если *выгрузка АРМ*, то запись из ArmWork удаляется

Если выгрузка последнего АРМ, где работал данный пользователь, то UserWork.Oper = 7

- Если *смена пользователя*, то завершение работы старого пользователя UserWork.Oper = 7 и начало работы нового UserWork.Oper = 6

3.5 МЕТОДЫ И СРЕДСТВА РАЗРАБОТКИ

3.5.1 БАЗА ДАННЫХ

В качестве БД используется PostgrePro

1. Т.к. в рамках одного подключения к серверу можно обращаться к данным только одной базы данных, указанной при установлении соединения, необходимо внести изменения в организацию БД. То, что было БД в SQL, в PostGreS станет схемой. Т.е. создана БД с именем «ASUST», а в ней созданы схемы «kcmod», «citns», «citnv» и т.д.

2. В качестве оболочки (средства работы с БД при разработке) используется я приложение DBeaver, его можно считать упрощенной версией SQL Management Studio.

3.5.2 ПЛАНИРОВЩИК (ОБРАБОТКА СООБЩЕНИЙ)

Написан на языке C++. Среда разработки Microsoft Visual Studio Enterprise 2019. Имеет в составе обработчики сообщений, написанные на языках C++, C#(NetCore), Java (стыковка с C++ написана.). Обработка конкретных сообщений реализуется в динамически подключаемых библиотеках. Т.Е. есть три компонента, которые предоставляют некоторые сервисные функции, например, доступ к НСИ, форматный контроль сообщений, рассылка сообщений другим абонентам, и загружают библиотеки с обработкой конкретных сообщений. Обработчики сообщений могут быть загружены в параллельном и последовательном режиме и в любых количествах.

3.5.3 WEB АРМ (ПРИЛОЖЕНИЕ АСУ ЖДП WEB)

Основное приложение системы – АСУ ЖДП WEB. Состоит из стандартного каркаса(оболочки) и набора прикладных страниц. Любой АРМ строится на основе общего каркаса. Страницы АРМа встроены в мастер-страницу оболочки. Т.е. имеют общую шапку (header) и нижний колонтитул (footer).

Реализована возможность WEBAPM, в рамках взаимодействия с ЦТО, иметь свой автоответ и логическое имя. Это позволяет WEB АРМ регистрироваться в системе, отправлять сообщения, получать квитанции и уведомления. Эти функции реализованы в рамках приложения ASUSTWEB.

АСУ ЖДП WEB написан на языках C# и JavaScript.В качестве платформа для разработки на момент написания документа используется .Net 5.0

Для соединения с PostgreSQL используются пакеты NpgSql и Dapper

В качестве доступных фреймворков используются jQuery

Для обмена сообщениями браузера с серверной частью используется SignalR

Отработано размещение приложения на сервере приложений Nginx

3.5.4 ЦЕНТРАЛЬНАЯ ТО ДЛЯ LINUX

НАЗНАЧЕНИЕ: работа в качестве центральной телеобработки как под Windows так и под Linux

ЯЗЫК ПРОГРАММИРОВАНИЯ: C# NET CORE 3.1 (хотя, возможно, достаточно меньшей версии)

Реализована в виде библиотеки классов. Стартовый класс запускается сторонним приложением (на момент - сделано консольное приложение для запуска ЦТО).

При старте, ЦТО загружает параметры работы (место хранения очередей, логов, имя таблицы с абонентами, список адресов "админов", число дней хранения логов и сообщений, списки кодов сообщений в/из АСОУП для пересылки в СП) из файла настроек.

Очереди абонентов хранятся в виде файлов. По 2 файла на каждый день хранения для каждого абонента. Один файл - массив структур описателей сообщений, второй - собственно сообщения. Файл создаётся размером с запасом на день. Размер вычисляется на основе статистики по предыдущим дням. По окончании суток файлы обрезаются под фактически занятое место. Пустая очередь или с истёкшим лимитом времени хранения удаляется.

Атрибуты Autor Abonent Prefix хранятся в UTF-8, поэтому, в байтах под них зарезервировано в 2 раза больше указанного количества. Предельное число символов получается зависящим от самих символов: чисто латиницы может быть в 2 раза больше.

NetTo.DLL доработана для поддержки работы с SenderID, ReplyID и расширенным Code;

На момент, ЦТО поддерживает 2 типа абонентов: STDP клиент и STDP сервер. Сама, соответственно, может выступать в качестве STDP клиента и STDP сервера. Список абонентов загружается из PG из таблицы citnv.armusernet. Для удобства ведения списка абонентов, абоненты с номером группы "0" - считаются абонентами с индивидуальными подключениями так же как и абоненты с уникальным номером группы. Абоненты с одинаковой группой (но не 0) - считаются находящимися в одной группе и очередь ведётся на 1-го абонента в группе.

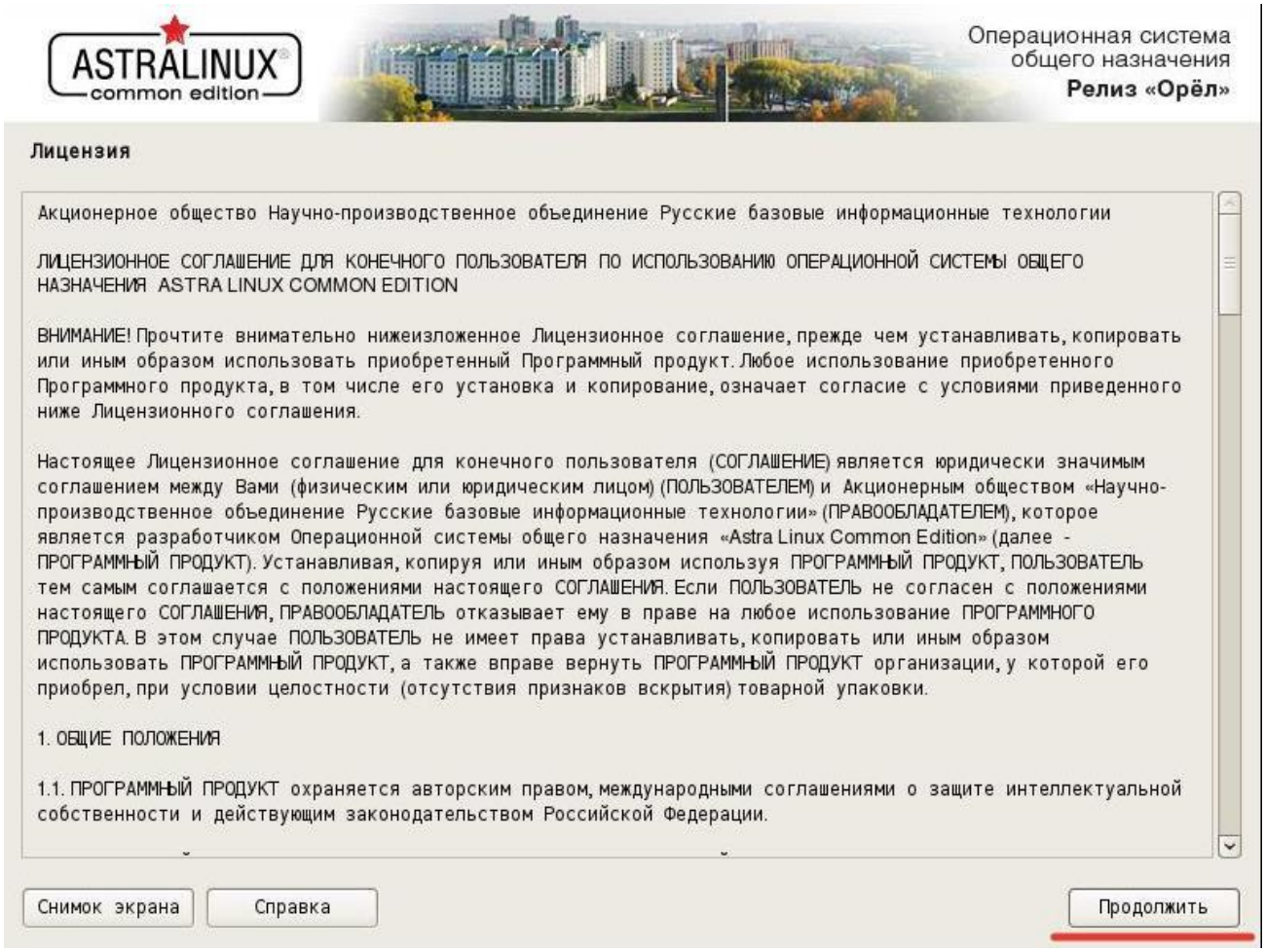
Для поддержки создания кеша сообщений на приём в ТО СП, ЦТО поддерживает опережающую отправку сообщений, не дожидаясь обработки предыдущих.

Просмотр состояния каналов, очередей, поиск сообщений и т.п. осуществляется управляющими командами, передаваемыми через TCP сервер, выполняющий роль единой точки подключения как STDP клиентов так и Консоли. Чистка очередей разрешена только администраторам, адреса которых внесены в настроечный файл (если указаны, иначе - всем).

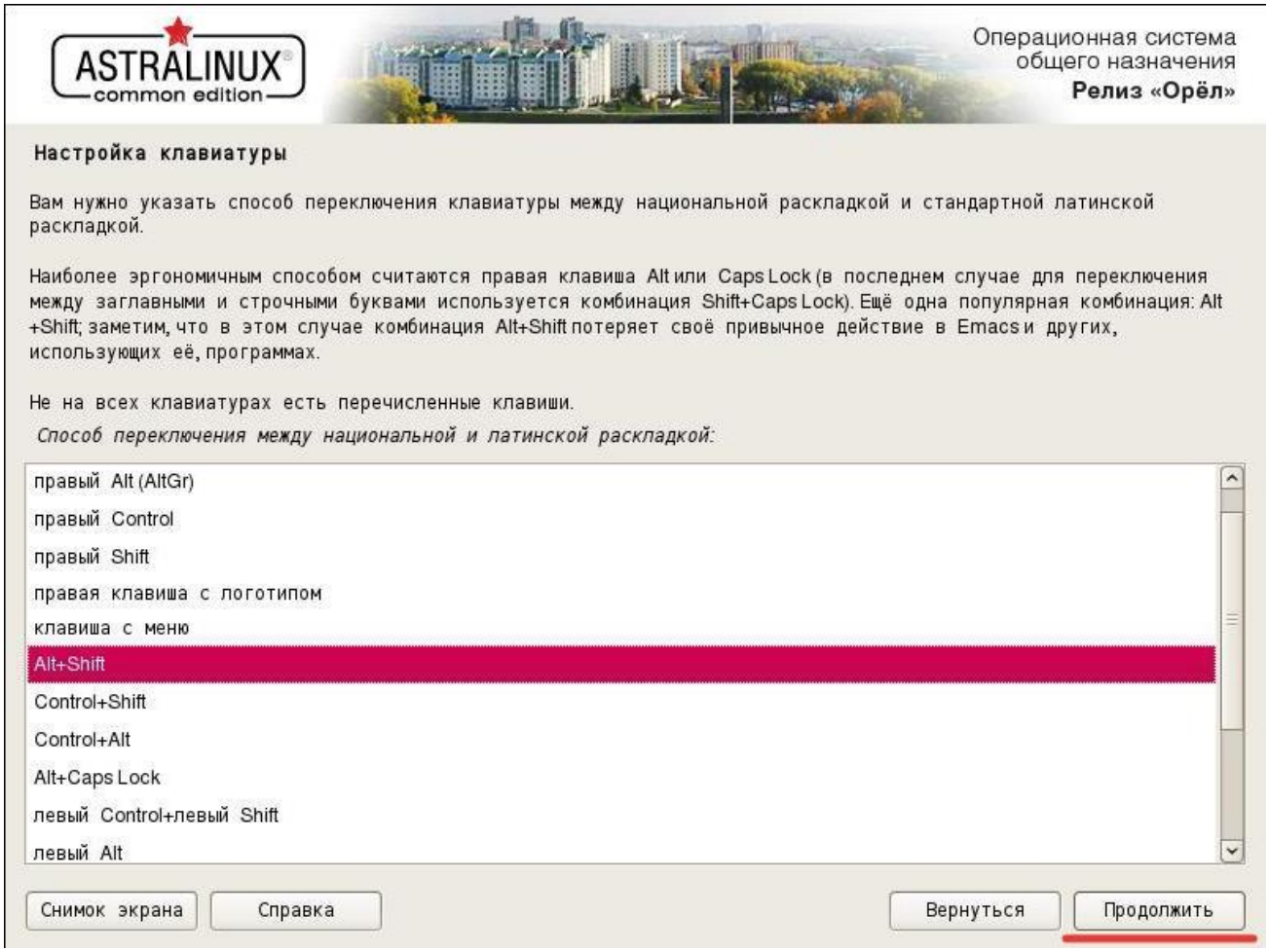
4. УСТАНОВКА ASTRA LINUX

1. Графическая установка

2. ПРОДОЛЖИТЬ

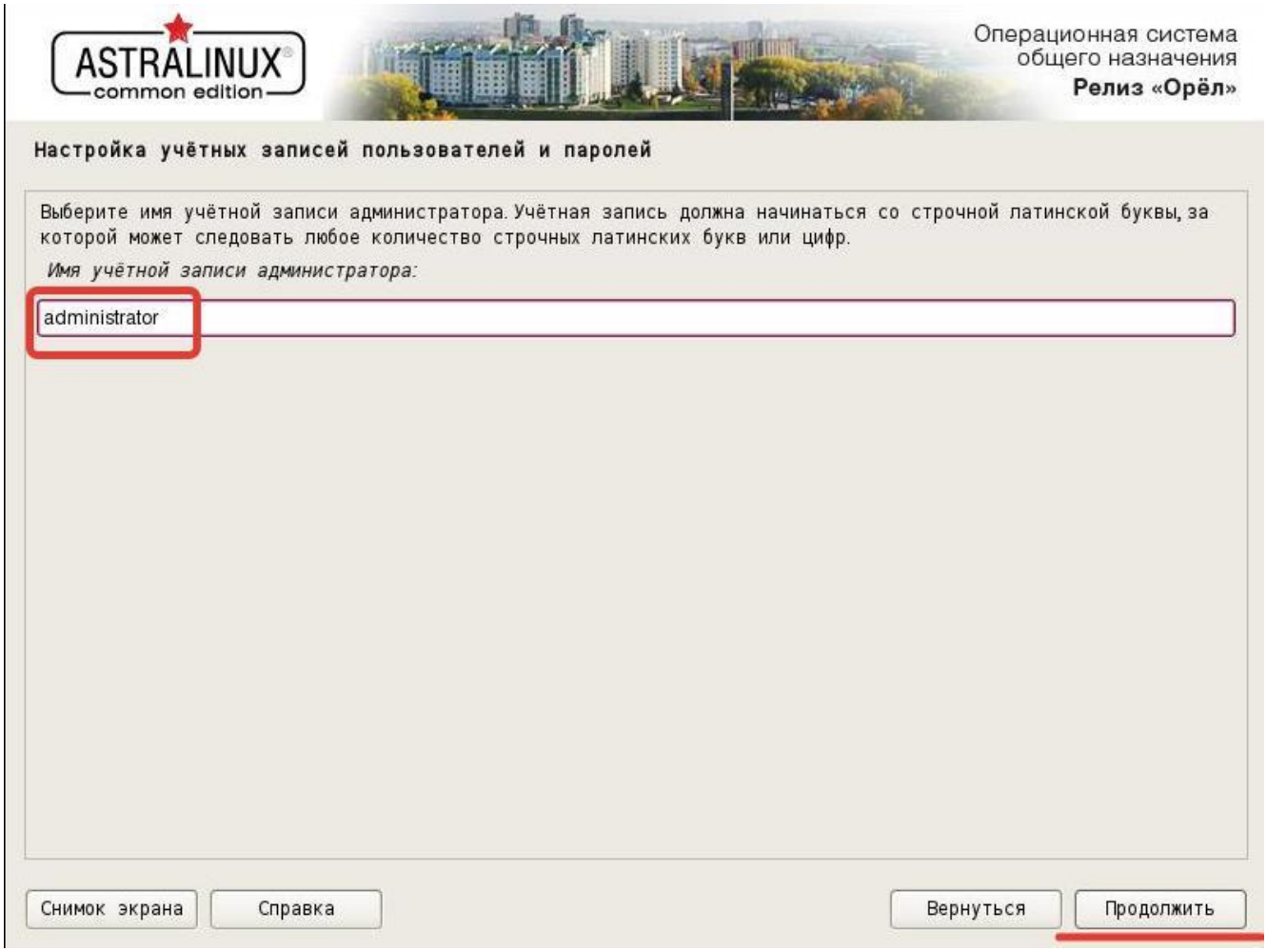


3. Выбор клавиш смены раскладки. ПРОДОЛЖИТЬ.

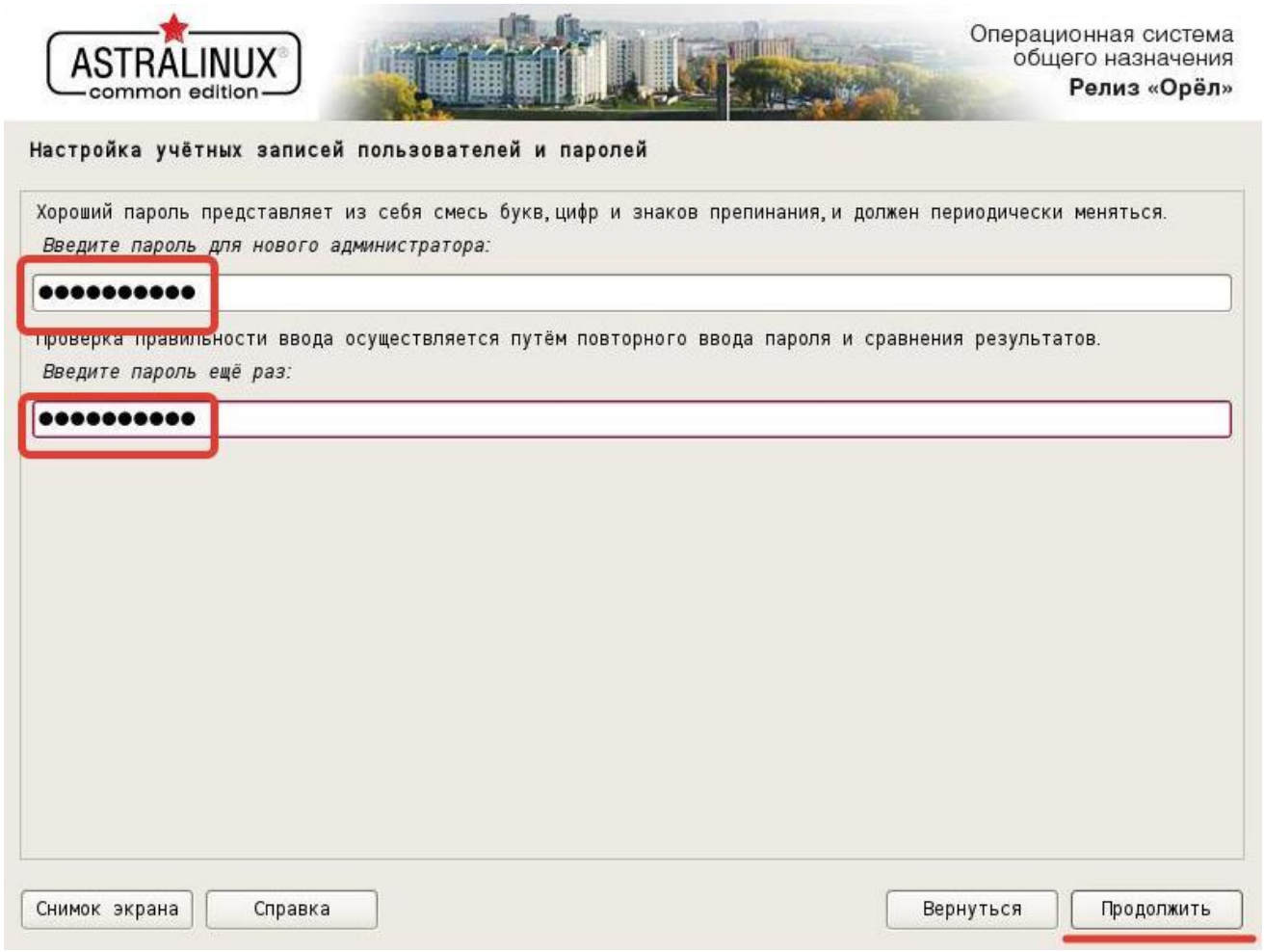


4. Ввести ИМЯ компьютера. ПРОДОЛЖИТЬ

5. Ввести ИМЯ учетной записи. ПРОДОЛЖИТЬ.

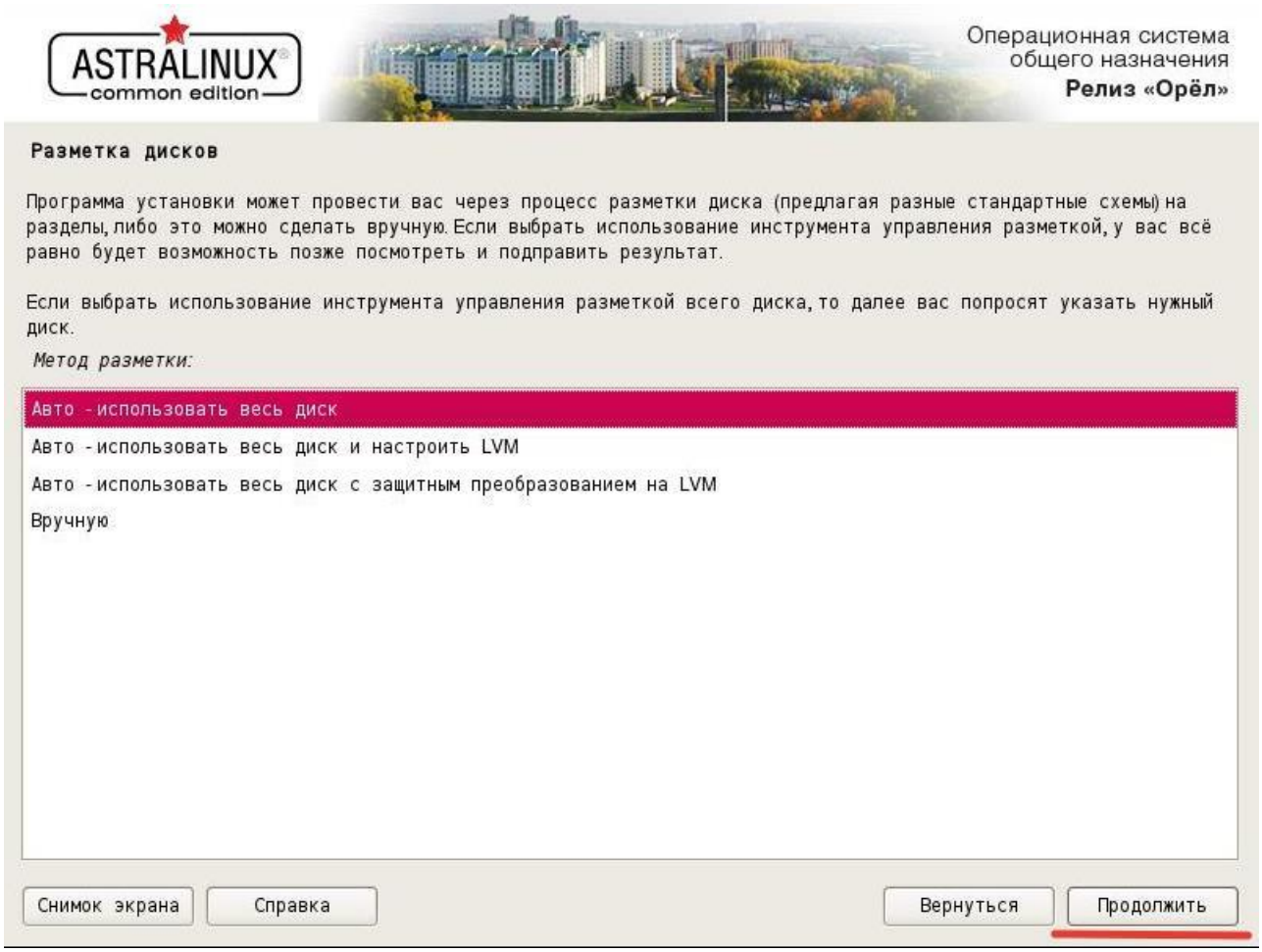


6. Ввести пароль Учетной записи. ПРОДОЛЖИТЬ.

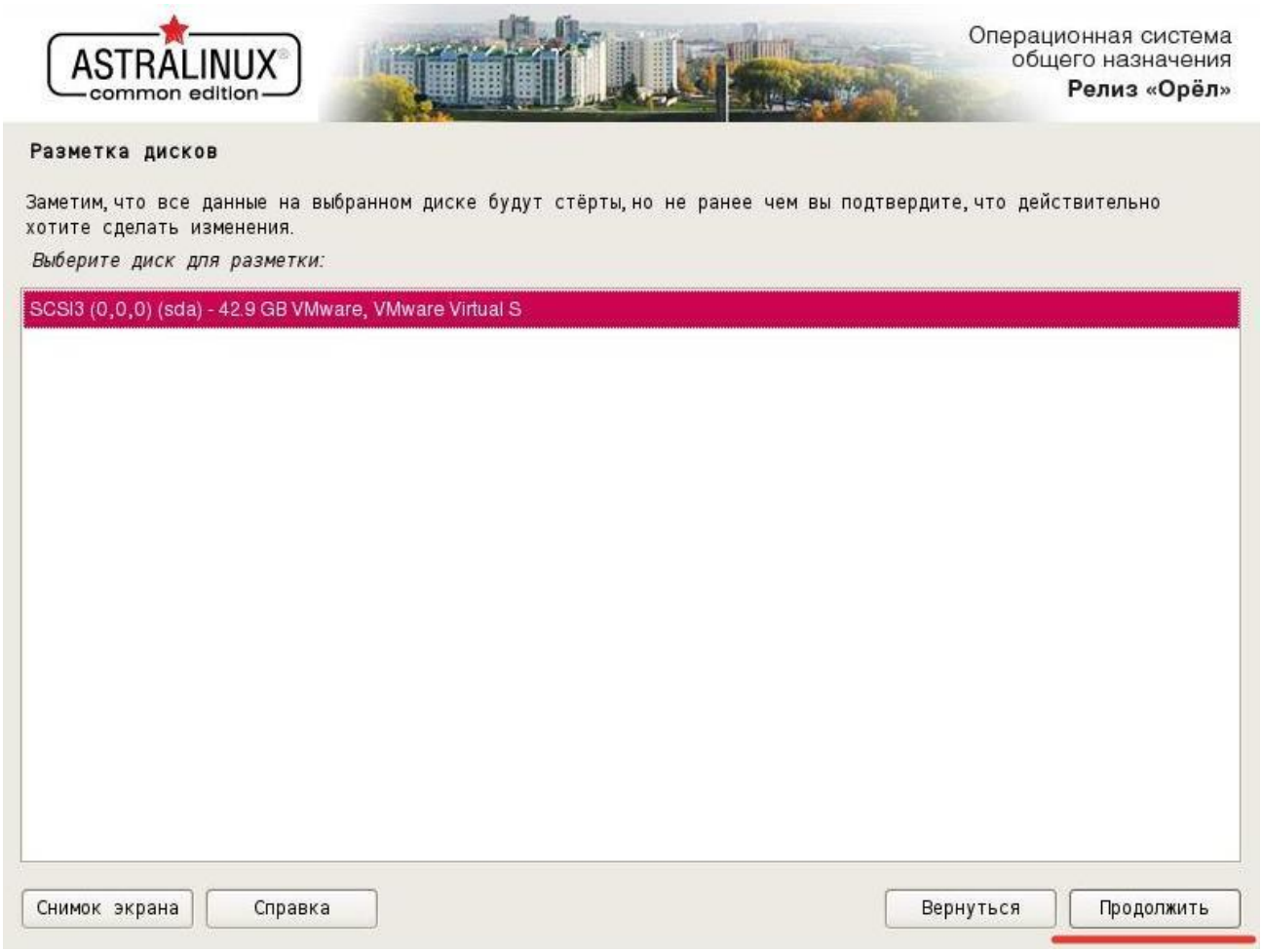


7. Выбор часового пояса. ПРОДОЛЖИТЬ.

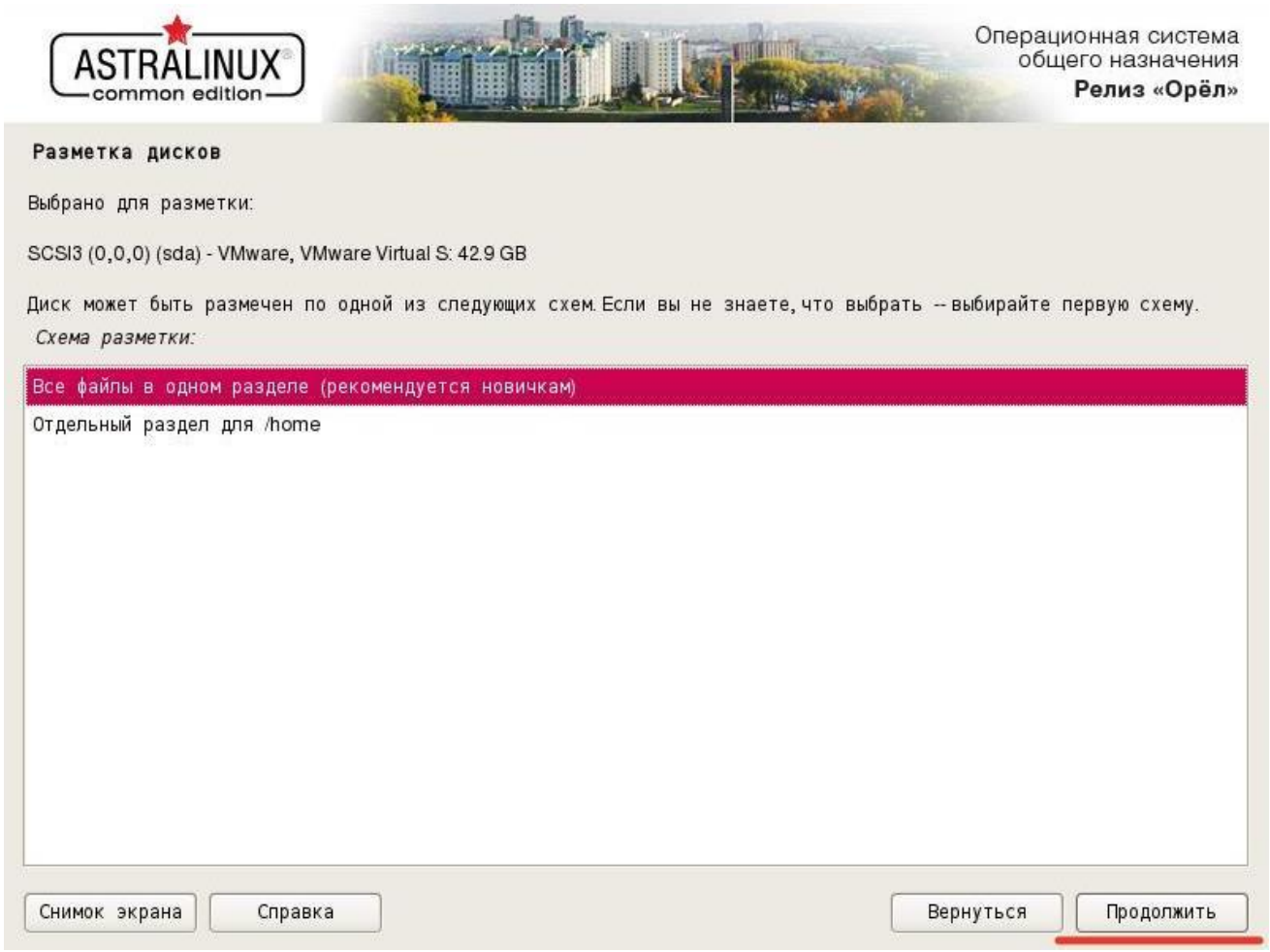
8. Выбрать *Авто – использовать весть диск*. ПРОДОЛЖИТЬ.



9. Выбрать диск для установки Операционной системы. ПРОДОЛЖИТЬ.

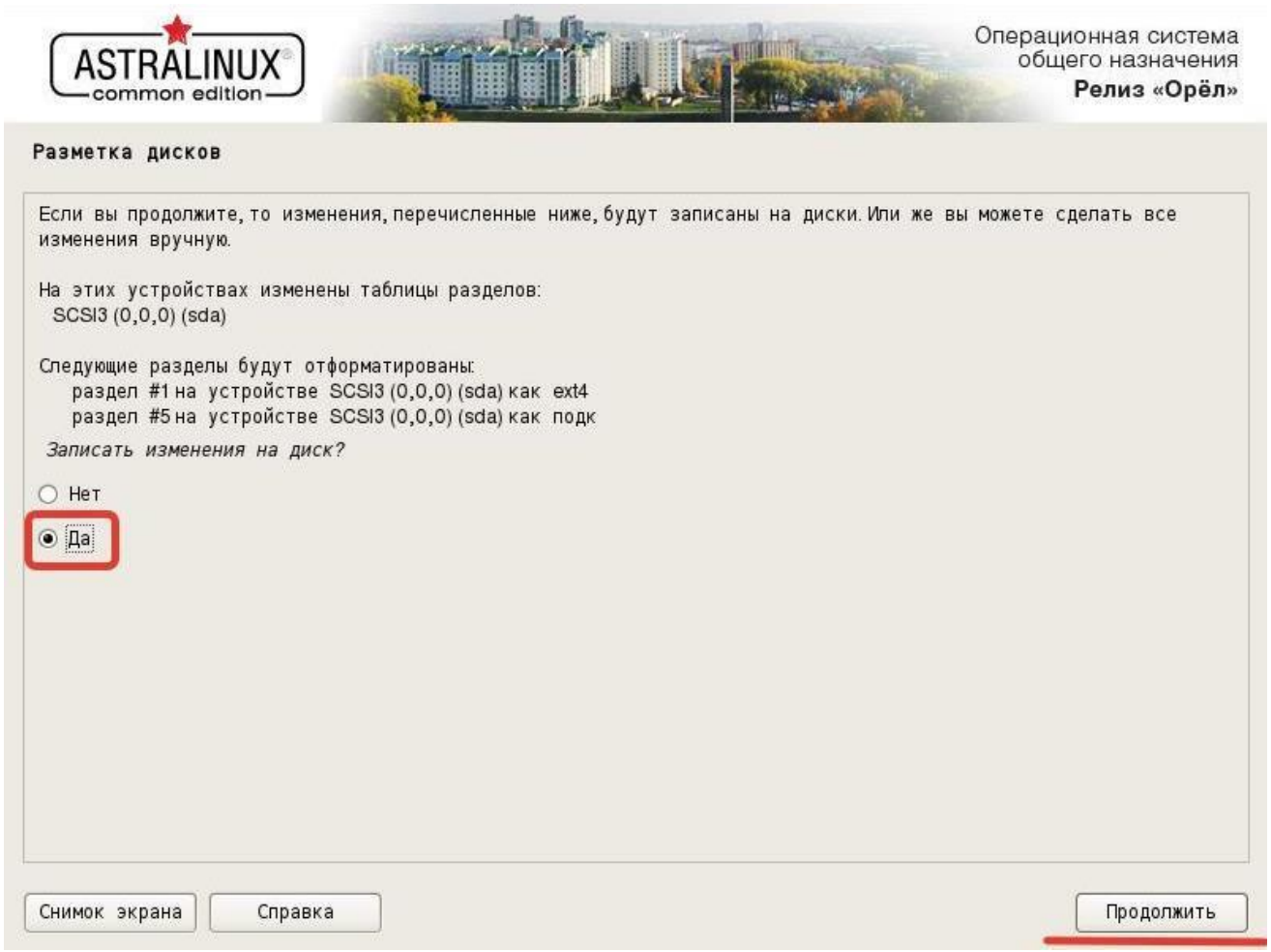


10. Выбрать раздел для установки. ПРОДОЛЖИТЬ.

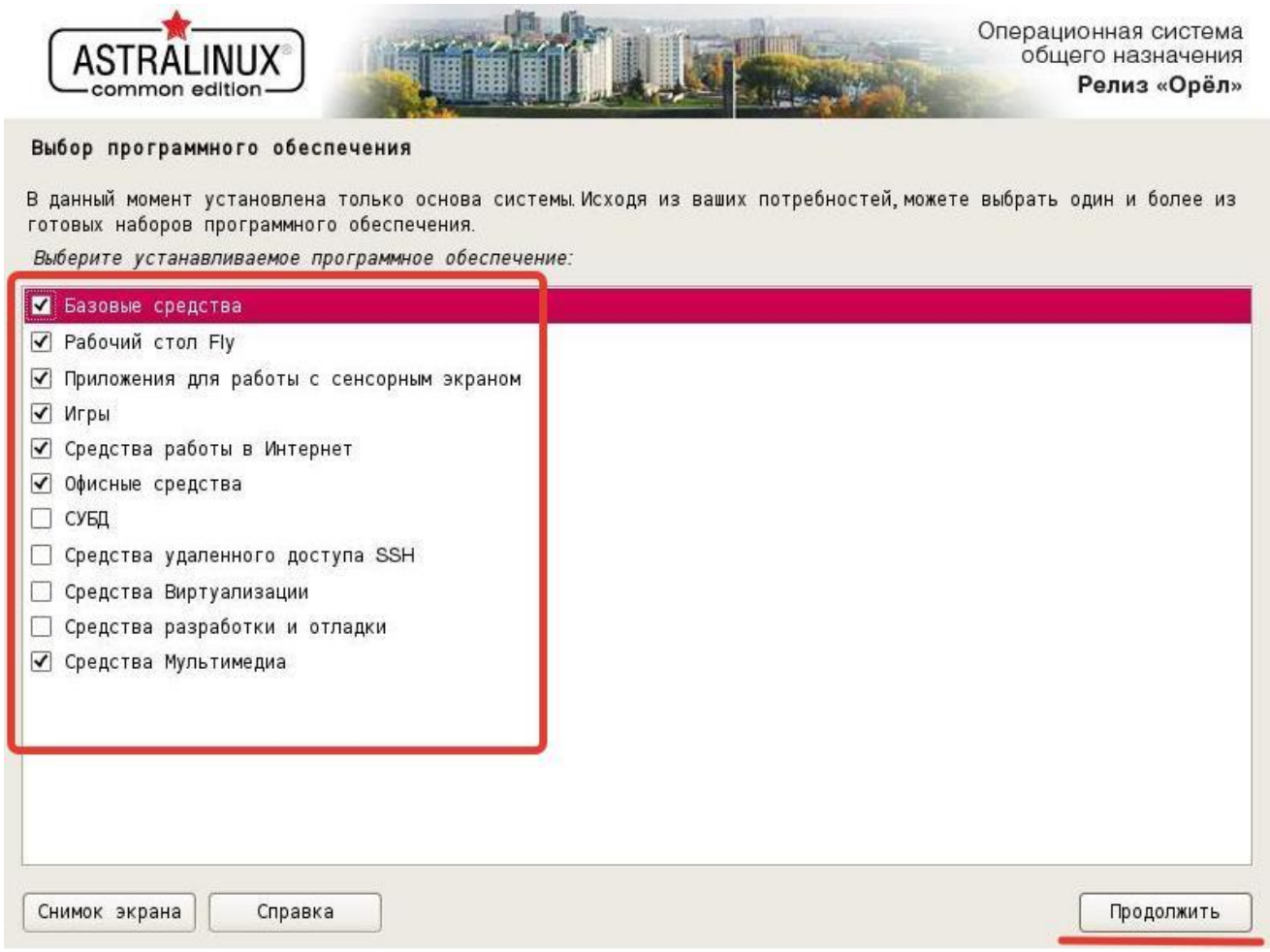


11. Выбрать Закончить разметку и записать изменения на диск. ГОТОВО.

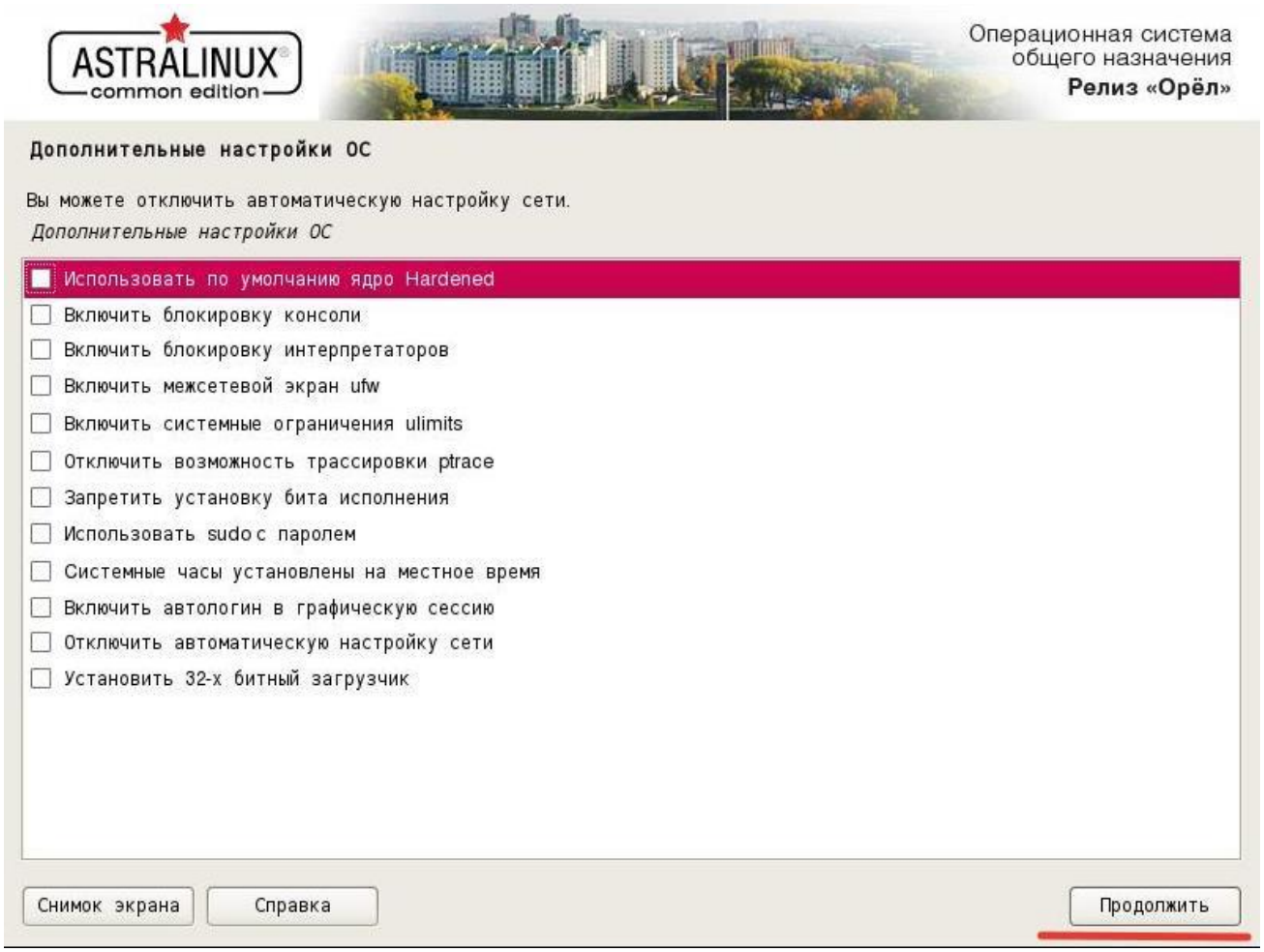
12. Поставить галку на ДА. ПРОДОЛЖИТЬ.



13. Выбрать ПО для установки. После можно установить необходимое из депозитариев. ПРОДОЛЖИТЬ.

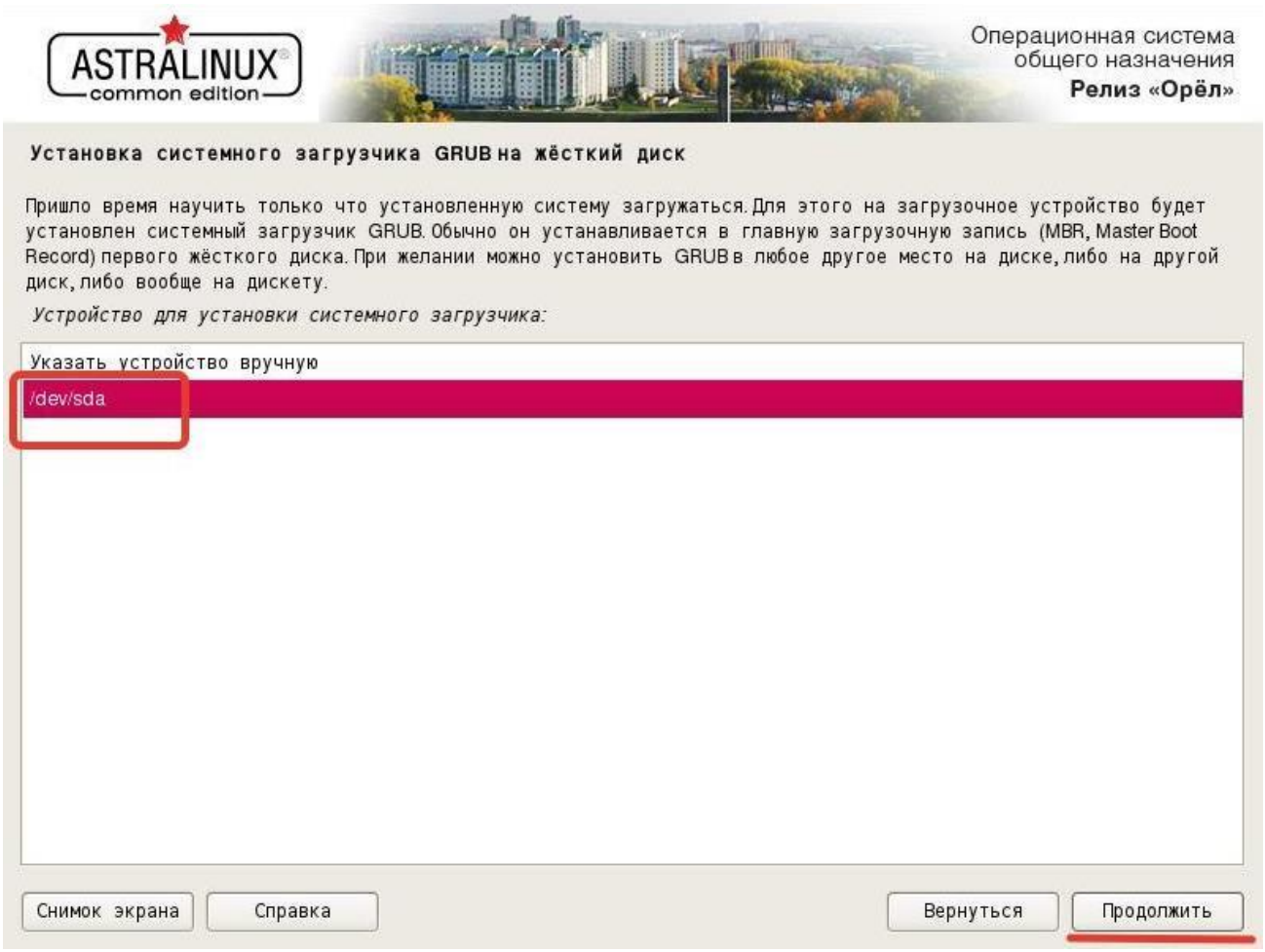


14. Выбрать необходимые дополнительные настройки. ПРОДОЛЖИТЬ.

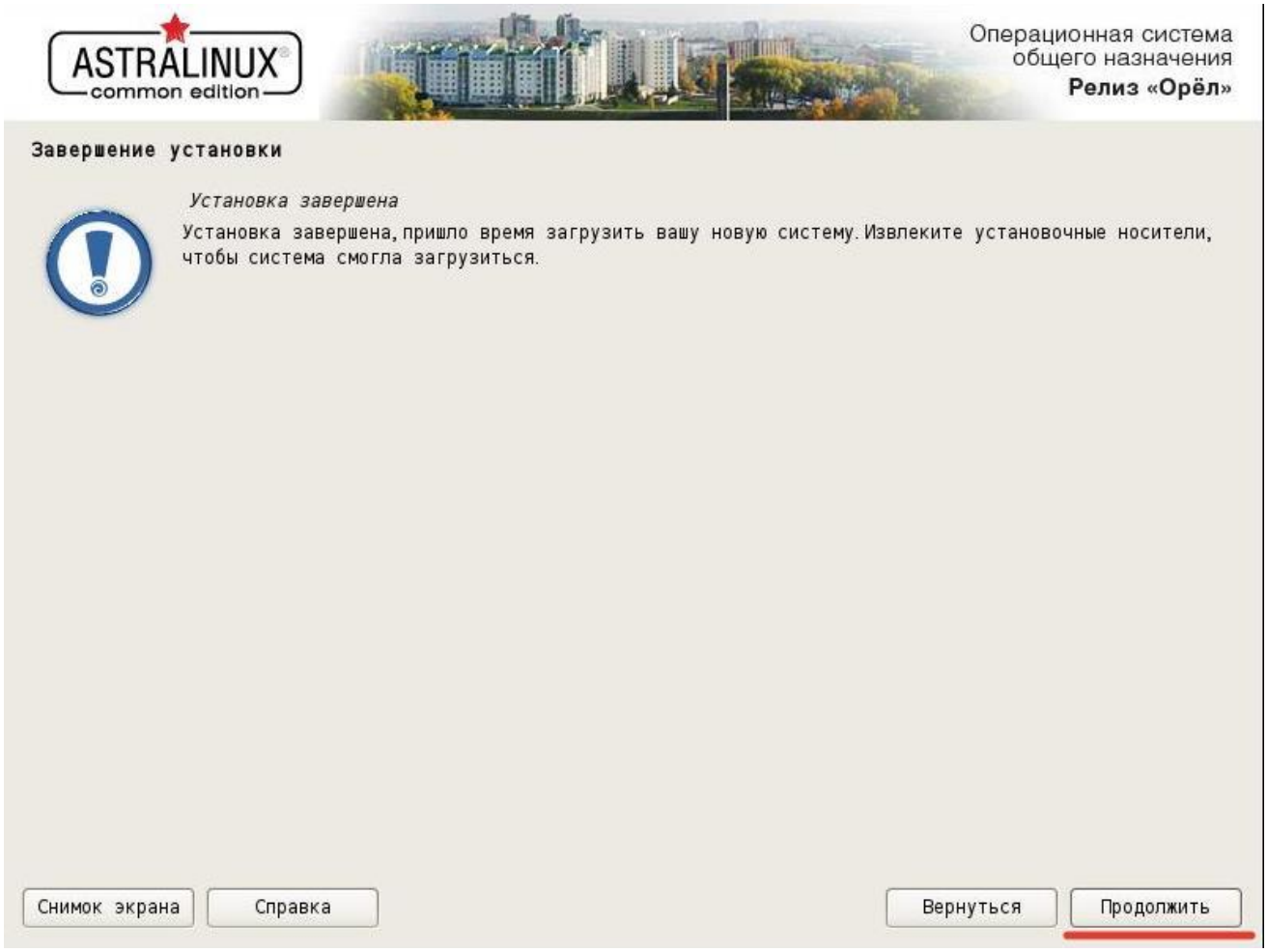


15. Поставить галку на ДА. ПРОДОЛЖИТЬ.

16. Выбрать раздел для загрузчика GRUB. ПРОДОЛЖИТЬ.



17. ПРОДОЛЖИТЬ.



18.Ввод Логин/Пароль указанный при установке.


Вход в astra
вторник, 30 июля 2024 г, 14:53:37 MSK
Администратор
administrator
Имя:
administrator
Пароль:
Тип сессии
Меню
En

19.ГОТОВО

5. ПОДГОТОВКА К ЗАПУСКУ СИСТЕМЫ И ИНСТАЛЛЯЦИЯ

5.1 БД POSTGRES PRO. ОБОСНОВАНИЕ ВЫБОРА РЕДАКЦИИ

Цена редакции Standard примерно в три раза меньше цены Enterprise.

Лицензия СУБД Postgres Pro AC Standard на 1 ядро RISC - 340 000 р

+ 80 000р - поддержка на год

Лицензия СУБД Postgres Pro AC Enterprise на 1 ядро RISC - 1 250 000 р

+ 250 000р.- поддержка на год

5.1.1 ОСОБЫЕ ВОЗМОЖНОСТИ POSTGRES PRO ENTERPRISE

Симметричный отказоустойчивый кластер (мультимастер)

Оптимизированное секционирование таблиц

Сжатие данных на уровне блоков

Инкрементальное резервное копирование

Использование механизмов машинного обучения при планировании запросов

Встроенный пулер соединений

Полнотекстовый поиск с ранжированием по релевантности

Автоматическая компиляция и планирование запросов

5.1.2 ПОЧЕМУ ВЫБИРАЮТ СУБД POSTGRES PRO ENTERPRISE?

Производительность

До двух раз выше по сравнению с PostgreSQL.

Безопасность

Сертификат ФСТЭК. Входит в Единый реестр Минкомсвязи. Совместим с СКЗИ «Крипто БД 2.0»

Надежность

Отказоустойчивый кластер мультимастер сохранит данные при отказе любого узла.

Встроенный пулер соединений

Встроенный пулер соединений с поддержкой сессий, подготовленных запросов и временных таблиц, позволяющий увеличивать на порядок количество одновременно работающих с базой данных пользователей. Доступен как встроенный пул соединений с поддержкой сессий, так и сторонний (pgbouncer).

Автоматическая подготовка SQL-запросов

Автоматическая компиляция и планирование исполнения SQL-запросов сокращает расходы ресурсов за счет повторного использования ранее созданных планов запросов. Добавлена поддержка JIT-компиляции запросов.

Ускоренный индексный поиск

Ускорено создание индексов за счет оптимизации алгоритмов работы с разделяемой памятью. Добавлены покрывающие индексы, содержащие дополнительные поля для ускорения выполнения запросов. Ускорен индексный поиск по JSON. Улучшена производительность индексного поиска и снижены блокировки при использовании индексов. Ускорено освобождение места при VACUUM.

Оптимизатор SQL-запросов

Более точный учет стоимости сравнения при сортировке. Автоматический оптимальный выбор последовательности группировки данных. Использование данных индексов для оценки количества результатов при соединении таблиц. Автоматическое удаление самосоединений таблиц, если оно не влияет на результат запросов. Улучшение оценки селективности для логических данных и для неточного сравнения. Улучшение работы оптимизатора с конструкциями [NOT] EXISTS и HAVING.

Резюме.

На первые несколько лет лучше приобретать редакцию Standard, т.к. перевод функционала будет поэтапным и нагрузка на БД также будет возрастать постепенно. К моменту окончания работ потребуется перейти на работу с редакцией Enterprise.

5.2 ОБНОВЛЕНИЕ ОС (ASTRA)

Для установки обновления с использованием интернет-репозитория Astra Linux CE необходимо выполнить следующие действия:

Подключить интернет-репозиторий Astra-Linux CE (если ранее он не был подключен). Для этого в файл «/etc/apt/sources.list» добавить строку:

для протокола https:

deb https://dl.astralinux.ru/astra/frozen/2.12_x86-64/2.12.45/repository/ orel main contrib non-free

для протокола http:

deb http://dl.astralinux.ru/astra/frozen/2.12_x86-64/2.12.45/repository/ orel main contrib non-free

Для использования сетевых репозиториев работающих по протоколу https установить пакеты apt-transport-https и ca-certificates

sudo apt install apt-transport-https ca-certificates

Выполнить синхронизацию файлов описаний пакетов с их источником:

sudo apt update

Установить обновления

sudo apt dist-upgrade

После выполнения обновления перезагрузить систему.

Если необходимо использовать предыдущие версии репозиториев, то в файле «/etc/apt/sources.list» необходимо удалить (закомментировать) строку с

описанием текущего актуального репозитория и добавить строку указав в ней необходимую версию ОС. Пример:

deb https://dl.astralinux.ru/astra/frozen/2.12_x86-64/2.12.<номер_обновления>/repository/ orel main contrib non-free

После этого выполняем обновление как описано выше.

На текущий момент поддерживаются следующие версии обновлений Astra Linux CE:

2.12.13

2.12.14

2.12.21

2.12.22

2.12.29

2.12.40

2.12.42

2.12.43

2.12.44

5.3 УСТАНОВКА POSTGRESQL

Предпочтительным вариантом является использование готовых пакетов, так как в этом случае получается понятная, поддерживаемая и легко обновляемая установка.

В пакетные репозитории некоторых систем уже входит PostgreSQL, но, обычно, не самой последней версии. Актуальная версия может быть взята из репозитория PostgreSQL Global Development Group (PGDG): «https://www.postgresql.org/download/».

Для автоматического добавления репозитория PostgreSQL Global Development Group необходимо выполнить следующие действия:

Установить пакет postgresql-common:

sudo apt install -y postgresql-common

Выполнить скрипт создания файла репозитория pgdg.list:

sudo /usr/share/postgresql-common/pgdg/apt.postgresql.org.sh stretch

После этого обновляем список пакетов и устанавливаем postgresql актуальной версии.

sudo apt update

sudo apt install postgresql-16

Дале необходимо выполнить инициализацию БД командой initdb. Она создаёт структуру хранения, системные таблицы и важные файлы конфигурации при первом запуске Postgresql. Выполнять команду необходимо от имени пользователя postgres. Параметр –D определяет каталог, в котором будут храниться данные. Если параметр –D не указан каталог определяется при помощи переменной среды $PGDATA

Пример комманды:

initdb –D /var/1ib/pgsql/16/data

5.4 УСТАНОВКА РАСШИРЕНИЙ POSTGRESQL

Расширения поставляемые с Postgresql входят в пакет contrib и доступны после установки. Расширения не входящие в contrib-пакет необходимо скомпилировать и добавить в директории Postgres в соответствии с рекомендациями разработчика расширения.

Для просмотра списка доступных расширений в БД необходимо выполнить запрос:

*select * from pg_available_extensions;*

Для установки расширения в БД необходимо выполнить запрос:

CREATE EXTENSION ext_name;

Для удаления расширения из БД необходимо выполнить запрос:

DROP EXTENSION ext_name;

5.5 РЕЗЕРВНОЕ КОПИРОВАНИЕ

Логическая резервная копия – это набор команд SQL восстанавливающий кластер (БД или отдельный объект) с нуля.

pg_dump – утилита для создания создания резервных копий PostgreSQL. Она позволяет создавать целостные резервные копии отдельных баз данных, не препятствуя работе с базой данных.

Создание резервной копии локальной базы данных (параметр –U не обязательный):

pg_dump –U username dbname > /tmp/dbname.sql

Создание резервной копии отдельной таблицы локальной базы данных:

pg_dump -t example_table dbname > /tmp/dbname_example_table.sql

pg_dumpall – это утилита для создания резервных копий PostgreSQL, позволяет создать резервную копию всех баз данных внутри одного инстанса PostgreSQL, также выгружает глобальные объекты, общие для всех баз данных, такие как роли и табличные пространства.

Создание резервной копии локальной базы данных (параметр –U не обязательный):

pg_dumpall > -U username /tmp/dump.sql

Результатом работы утилит pg_dump и pg_dumpall является скрипт для sql, для восстановления резервной копии можно воспользоваться консольной утилитой psql. Пример использования утилиты psql:

psql –d dbname < /tmp/dbname.sql

5.6 МАСШТАБИРОВАНИЕ АСУ ЖДП ВЕБ

В качестве сервера приложений для системы АСУ ЖДП ВЕБ выбран сервер Nginx, который позволяет конфигурировать горизонтальное масштабирование системы и обеспечивать балансировку нагрузки, а также отказоустойчивость системы (рисунок 14).

Рисунок 14

В общем случае система АСУ ЖДП ВЕБ развёртывается на неограниченном количестве серверов. Клиент обращается с запросом к балансировщику, в котором сконфигурировано перенаправление запроса на сервера, с учётом выбранного алгоритма перенаправления. Таким образом распределяется нагрузка между серверами. Также при отказе одного из серверов запросы перенаправляются на работающие сервера.

В частном случае допустимо развёртывание балансировщика на одном из рабочих серверов.

5.7 УСТАНОВКА РАСШИРЕНИЙ ДЛЯ ОБРАЩЕНИЙ К ВНЕШНИМ СЕРВЕРАМ

Для обращения к данным находящимся на внешнем сервере Postgres используется расширение postgres_fdw. Этот модуль уже по умолчанию входит состав Postgres.

Для просмотра доступных расширений:

select * from pg_available_extensions;

Для просмотра установленных расширений:

```
select * from pg_available_extensions where installed_version is not null;
```

Если расширение не установлено – устанавливаем его:

```
create extension postgres_fdw;
```

Порядок действий следующий:

Создаем подключение к стороннему серверу:

```
CREATE SERVER foreign_astra_test2
        FOREIGN DATA WRAPPER postgres_fdw
        OPTIONS (host '172.17.129.103', port '5432', dbname 'Test');
```

Создаем сопоставление пользователей для нашего подключения

```
CREATE USER MAPPING FOR postgres
        SERVER foreign_astra_test2
        OPTIONS (user 'postgres', password 'postgres');
```

Создаём стороннюю таблицу:

```
CREATE FOREIGN TABLE fdw_testtable
        ("TESTID" INT8, "MNKD" VARCHAR(9), "NMN" VARCHAR(50), "DSCR" VARCHAR(255))
        SERVER foreign_astra_test2
        OPTIONS (schema_name 'public', table_name 'TESTTABLE');
```

5.8 ЗАПУСК И ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА АСУ ЖДП PG

1. Развернуть БД. Выполнив SETUP или BACKUP БД.
2. Настроить удаленные сервера, согласно пункту 1 документа «Перекачка НСИ SQL- PG»

Документ описывает закачку НСИ в БД Postgres из действующей БД MS SQL АСУ ЖДП.

1. В citnv.nvServerNastr, Nastrid = 6, прописать код дороги. Он используется при расчете ИД Инфраобъекта.

2. sql01,sqlnv,sqlns для баз kcmod, citnv, citns:

   -ALTER SERVER sql01 OPTIONS (SET servername ‘10.240.3.74’);
   -ALTER SERVER sql01 OPTIONS (SET port ‘5142’);
   -ALTER USER MAPPING FOR public SERVER sql01 OPTIONS (SET password ‘pwd’);

   - ALTER SERVER sqlnv OPTIONS (SET servername ‘10.240.3.74’);
   - ALTER SERVER sqlnv OPTIONS (SET port ‘5142’);
   - ALTER USER MAPPING FOR public SERVER sqlnv OPTIONS (SET password ‘pwd’);

   - ALTER SERVER sqlns OPTIONS (SET servername ‘10.240.3.74’);
   - ALTER SERVER sqlns OPTIONS (SET port ‘5142’);
   - ALTER USER MAPPING FOR public SERVER sqlns OPTIONS (SET password ‘pwd’);

3. Перекачать НСИ согласно пунктам 2-4 документа «Перекачка НСИ SQl- PG»
   - *порядок первоначальной закачки НСИ:*
   1. **citns.psksmi_refresh_first_obj()** - процедура первоначального формирования Citns.InfraObj, Citns.InfraCountry, Citns.InfraDoroga, citns.InfraStan. InfraStan – сумма записей двух таблиц: fdw.nsstan и fdw.nsstandop.
   2. **citns.psksmi_refresh_first_nsstandop()-** процедура формирования citns.InfraStanDop на основе Citns.nsstanOsn_Dop.
   3. **citnv.psksmi_refresh_first_nvstan()** - процедура формирования CitNv.Infra Stan, Park, Put, Podx, Mesto на основе nv таблиц. Objid

ищем в citns.infrastan. Citnv.infrastan полностью заменяем: все удаляем и полная вставка.

- *процедуры синхронизации НСИ:*

4. **citns.psksmi_refresh_citns_infrastan()** - процедура добавления новых станций в citns.infrastan из fdw.nsstan и fdw.nsstandop
5. **citns.psksmi_refresh_citns()** - процедура принудительной синхронизации таблиц ns НСИ PG относительно НСИ SQL по списку citns.ListNsiNs.

4. Перекачать поездную, локомотивную и вагонную модель согласно документа «Перекачка поездной и вагонной модели»:

- Качаем всю дорожную модель.
- На время перекачки работа АСУ ЖДП останавливается.
- Архив не перекачиваем.
- Таблицу событий Event заполняем, создавая одно событие «Перекачка» с кодом 10.3
- Список таблиц для чистки:
  - PodxTrainStation
  - Train
  - TrainOper
  - TrainArxOper
  - TrainArxOsn
  - TrainArxRazl

  - Vagon
  - VagonCs
  - VagonBrak_prt

- Vagon_mest
- VagonOper
- VagonSl
- VagonArxOper
- VagonArxOsn
- VagonArxPasp
- VagonArxBrak
- VagonArxRzm
- VagonArxSl

- Lok_M
- Lok_Ms
- mSksIsInsertTrnArx
- mSksIsInsertVagArx
- mSksTrnArxTmp
- mSksVagArxTmp

- Event
- EventArx
- Journal
- Journal1
- JournalData

- Список таблиц для чтения из SQL:
  - PodxTrainStation
  - Train
  - TrainOper
  - Vagon
  - VagonCs

- VagonBrak_prt
- Vagon_mest
- VagonOper
- VagonSl
- Lok_M
- Lok_Ms

- Список таблиц для формирования после скачивания:
  - Event
  - EventArx
  - VagonArxOper
  - TrainArxOper

- Поезда перекачиваются все, вагоны 1 и 2 типа, в поездах и на местах подач. Перед началом закачки все таблицы по списку обнуляются.
- Процедура: **SELECT** kcmod.psksmi_refresh_first_model();

5. Установить ЦТО:

   Нижеописанные действия необходимо выполнять под тем пользователем под которым будет запускаться ЦТО либо давать ему необходимые права на каталоги или файлы.

   Создать каталог для программы и скопировать в него содержимое «\CIT-NEW\Свободное ПО\DISTRIB\СТО». Изменить установленный фреймворк на NET CORE 3.1 в файлах *.json, если вместо NET5 установлен NET CORE 3.1

Отредактировать файл settings.ini в части указания местоположения каталогов для очередей, логов и строки подключения к базе. Пример настроечного файла (кодировка UTF-8):

{
«КаталогЛогов»:»E:\\MyFiles.C\\CTO_CORE\\logs»,
«КаталогОчередей»:»E:\\MyFiles.C\\CTO_CORE\\Queue»,
«СписокАдресовАдминов»:»10.200.66.162,172.17.131.117»,
«History»:»8»,
«СтрокаПодключения»:»Host=172.17.131.171;Port=5432;Username=94ostgres;Password=CITdl160adm;Database=asust;»,
«КодыКопийСообщенийВАСОУП»:[208,449,1042,02,09,209,230,333,12,1353,1354,1397],
«КодыКопийСообщенийИЗАСОУП»:[208,449,1042,02,09,209,230,333,12,1353,1354,1397]
}

Минимально необходимое для запуска – Строка Подключения. Пользователь из этой строки должен иметь права на чтение таблиц ArmUserNet, Citnv.nvsendsoob, Citnv.nvip, citnv.nvservernastr, citnv.srvlogimps, kcmod.mtomessagequeue, kcmod.mtomessagequeue. Последние 2 ещё и на изменение.

Создать (изменить, если есть) в таблице абонентов(armusernet) строку(и) с STDP сервером(ами) к которым будут подключаться абоненты. Для STDP серверов важны параметры: Автоответ (13 символов, включая 2 пробела), используемый адрес(конкретный или 0), порт, признак включенности (enabled). Linetype=10, typearm=202. Пример:

```
SELECT * FROM citnv.armusernet WHERE linetype=10
```

idarm logname autoanswer linetype codepage
signpersonaltelesoob userdirin signrestorequeue signstoresoob
addres port description enabled groupingid typearm
56 stdptest (?? ???? ?)01 10 1 1 0 0 0 0 8886 ЦТО_PG: Сервер STDP 1 0 202

Прописать АРМЫ – клиенты STDP. Для STDP клиентов важны параметры: Логимя, Автоответ (13 символов, включая 2 пробела) , кодировка, период проверки канала в сек. Signpersonaltelesoob, контроллируемый адрес(конкретный или *), группа STDP(0 или конкретная), признак включенности (enabled). Linetype=11 или 14(обычно 14), typearm в зависимости от типа АРМа. Пример:

SELECT * FROM citnv.armusernet WHERE linetype in (11,14)

idarm logname autoanswer linetype codepage
signpersonaltelesoob userdirin signrestorequeue signstoresoob
addres port description enabled groupingid typearm
abonentvid esr nomzayvki mnkd
42 ps_osnudk (PS ???? U)DK 14 3 1 180 0 0 * 0 Планировщик Устинов 1 NULL 201 0 NULL NULL NULL
37 jok2_astra (?? ???? ?)37 14 3 1 180 0 0 * 0 Планировщик Голышев 1 0 201 0 0 NULL Jok
14 CHUB_TK_CHUB (?? ???? ?)14 14 3 1 180 0 0 CHUB-HP 1 АРМ ТК у Чубчева локальный 1 0 225 0 0 ТКЧуб
8 CHUB_DSP (?? ???? ?)07 14 3 1 180 0 0 CHUB-HP 1 АРМ ДСП у Чубчева локальный 1 0 204 0 0 ДСПЧуб
4 CHUB_RS_CHUB (?? ???? ?)04 14 3 1 180 0 0 CHUB-HP 1 АРМ РС у Чубчева локальный 1 0 205 0 0 РСЧубЛок
1 ASUSTWEB_CHUB (?? ???? ?)03 14 1 1 180 0 0 172.17.131.142 1 АСУ ЖДП ВЕБ на машине у Чубчева 1 0 200 0 0 АСУ ЖДПВЕБЧуб

6. Установить Планировщик:

Установить NET Core 3.1 или добавить в менеджере пакетов репозиторий, для автоматической установки этого пакета. Пример для debian 10
wget https://packages.microsoft.com/config/debian/10/packages-microsoft-prod.deb -O packages-microsoft-prod.deb
sudo dpkg –I packages-microsoft-prod.deb
rm packages-microsoft-prod.deb
sudo apt update

Установить пакет pscittrans.deb.

Для создания базы данных (если нужно) запустить /opt/cittrans/setup_ps/setup_ps.out

Скорректировать /opt/cittrans/sp_ksarm/SpService.ini, прописав IP адреса текущей машины и базы данных, логимя, под которым будет запускаться служба.
Создать каталог /opt/cittrans/sp_ksarm/SERVSOOB.XXX где XXX логимя из SpService.ini

Положить файл лицензии в каталог /opt/cittrans/sp_ksarm/SERVSOOB.XXX

В /opt/cittrans/sp_ksarm/SERVSOOB.XXX создать файл setup.net с содержимым
;Каталог для создания очередей
TELEDIR=/opt/cittrans/queue
;Признак сохранения старых логов: History=YES или число дней

History=7

;Каталог для логов

Logdir=/opt/cittrans/LOGS

;Размер очереди входящих сообщений в памяти [1-999]

InMessageQueueDept=100

Порядок запуска планировщика.

Для запуска планировщика необходимо настроить таблицы НСИ nvSrvComp, nvIP и три таблицы : nsSrvComp, nsSrvMsg, nsSrvMsgXML должны быть настроены разработчиками в ЦИТТРАНС

Настройка НСИ.

Таблица nsSrvComp.

Таблица описывает все возможные компоненты.

progId - программный идентификатор com-класса компонента

compId - идентификатор компонента (должен быть больше 0)

name - название компонента

compType - тип объекта(0-обычный,1-обрабатывает сообщения в отдельном потоке, 2-не принимает сообщения)

isMain - не принимать сообщения , если этоткомпонент не смог загрузиться

Пример:

progId	compId	name	compType	isMain
Server.pid	1	Сервер приложений	0	1
Accounts.pid	2	Отчетность	0	1

Spravka.pid 3 Справочная система 0 1

Таблица nsSrvMsg.

Сопоставляет код сообщения и компонент, который обрабатывает это сообщение.

msgNo - код сообщения. Должен указываться с ведущими нулями.

В любой позиции номера сообщения может стоять символ ‘?’, означающий любой знак.

compId - идентификатор компонента (из таблицы nsSrvComp)

orderNo – порядковый номер обработки сообщения.

maxTime – максимально допустимое время (в сек) обработки сообщения.

Таблица настраивается разработчиками.

Пример:

msgNo compId orderNo maxTime

???? 1 1 120

0777 3 1 240

Таблица nvSrvComp.

Описывает компоненты, загружаемые на данном объекте автоматизации.

compId - идентификатор компонента (из таблицы nsSrvComp)

name - название компонента(если при заполнении не введено, заносится из таблицы nsSrvComp)

Таблица настраивается для конкретной станции.

Пример:

compId name

---

3 Сервер приложений

3 Справочная система

Таблица nvSrvMsg.

Дает возможность изменить время обработки конкретного сообщения.

msgNo - код сообщения. Должен присутствовать в nsSrvMsg.

compId - идентификатор компонента (из таблицы nsSrvComp)

maxTime – максимально допустимое время (в сек) обработки сообщения.

Таблица nvSrvMsgDop.

Дает возможность изменить время обработки сообщения 504 и 1062 по коду операци.

msgNo - код сообщения. Должен присутствовать в nsSrvMsg.

msgNoUt - код операции в сообщении

maxTime – максимально допустимое время (в сек) обработки сообщения.

Таблица nsSrvMsgXML.

Сопоставляет код и xml сообщения по вхождению прописанной строки.

msgNo - код сообщения. Должен указываться с ведущими нулями.

messPart – строка

Таблица nvIP.

Необходимо прописать IP адрес SQL сервера, с которым работает ЦТО (где хранятся очереди ЦТО), описание заполнения смотри в «Таблица NvIP, описание серверов и баз АСУ ЖДП (бывшая nvSPOR).doc»

6 ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ

АСУ ЖДП работает в автоматическом режиме 24/7. В составе АСУ ЖДП есть встроенные средства мониторинга системы. Рассмотрим основные:

6.1 КОНСОЛЬ АСУ ЖДП

Назначение: просмотр таблиц событий (eventarx), журнала (journal), телеобработки (состояние, поиск сообщений …), пользователей (userinfo, UserWork).

Программа в папке «\CIT-NEW\Свободное ПО\АСУ ЖДП 1 этап работ\Консоль АСУ ЖДП» настроена на использование. NET7, поэтому, тем у кого стоит только NET5 необходимо скопировать все файлы к себе и заменить настройки файлами из папки «5.0». Проверить что у вас стоит можно в командном окне выполнив «dotnet –info» и посмотрев раздел «.NET runtimes installed». Интересуют строки на « Microsoft.NETCore.App …»

…

Microsoft.NETCore.App 5.0.17 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 6.0.15 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]
Microsoft.NETCore.App 7.0.4 [C:\Program Files\dotnet\shared\Microsoft.NETCore.App]

…

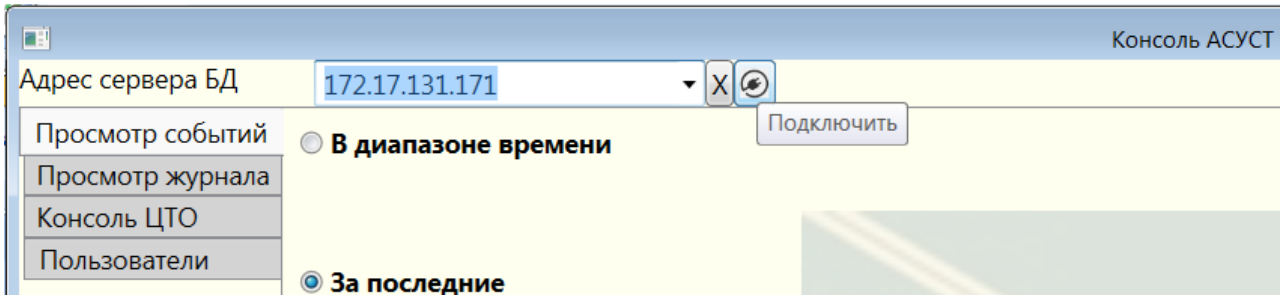


Рисунок 15

При запуске, программа автоматически подключается к последнему серверу БД. Для управления списком известных серверов предназначено поле ввода адреса сервера и кнопки подключить и удалить текущий (рисунок 15).

Просмотр событий и журнала предполагает ввод фильтров, после чего нажимается кнопка «найти» (рисунок 16).

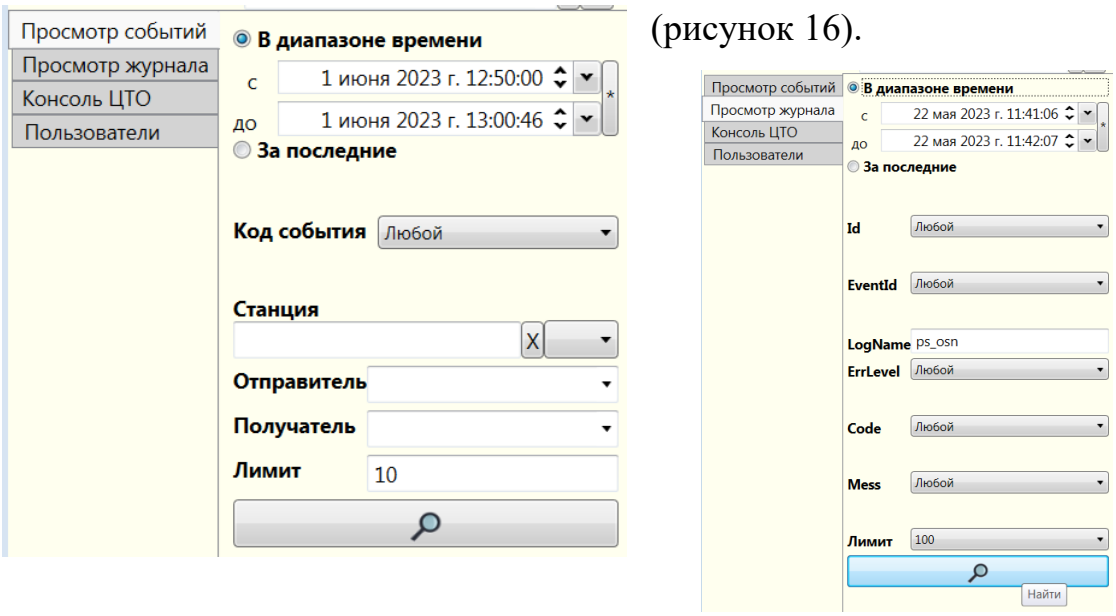


Рисунок 16

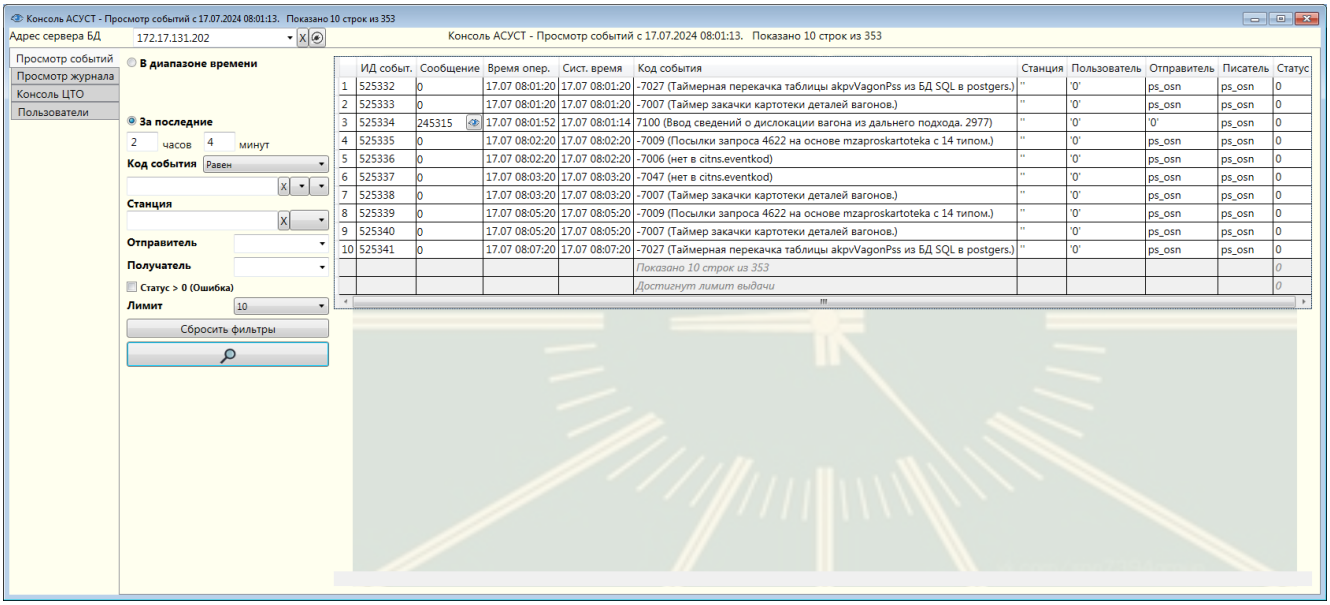


Рисунок 17

Если в колонке «Сообщения» есть кнопка – можно посмотреть сообщение, инициировавшее событие (рисунок 17).

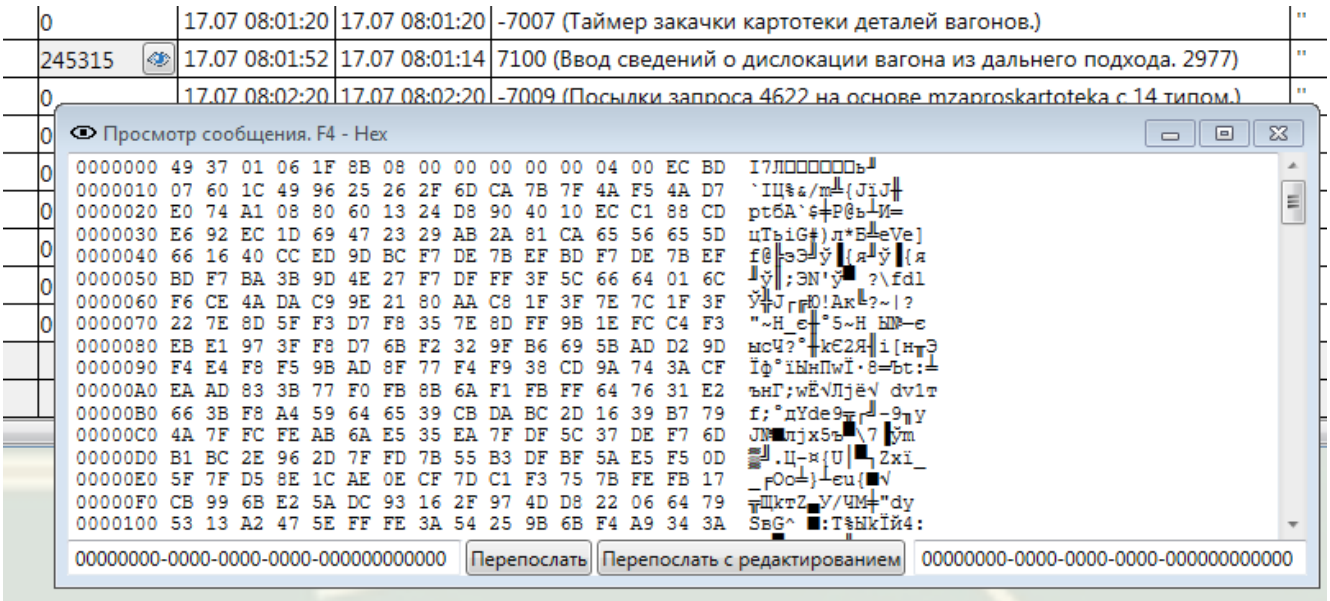


Рисунок 18

Двойной щелчок по событию открывает в нижней части журнал по этому событию (рисунок 18)

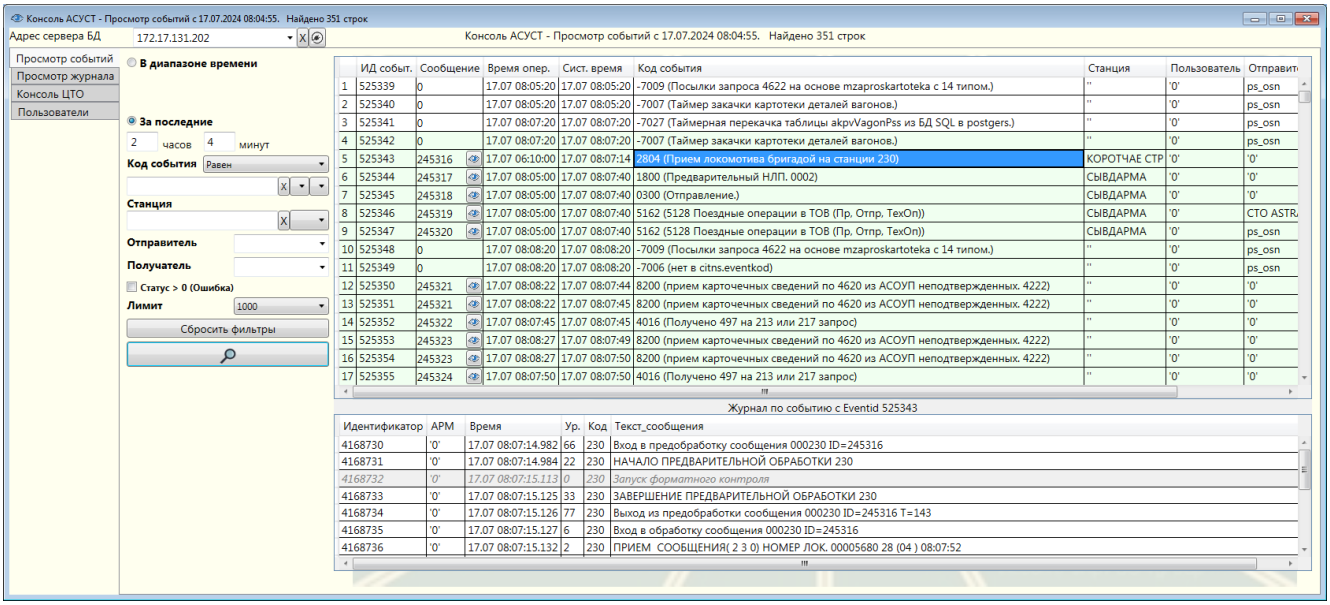


Рисунок 19

В вывод журнала включаются все записи с ID этого события и без ID события но с тем же кодом между первой и последней записью по этому ID события (рисунок 19).

Локальное меню строк позволяет посмотреть данные по событию из разных таблиц и добавлять условия в фильтр событий (рисунок 20).

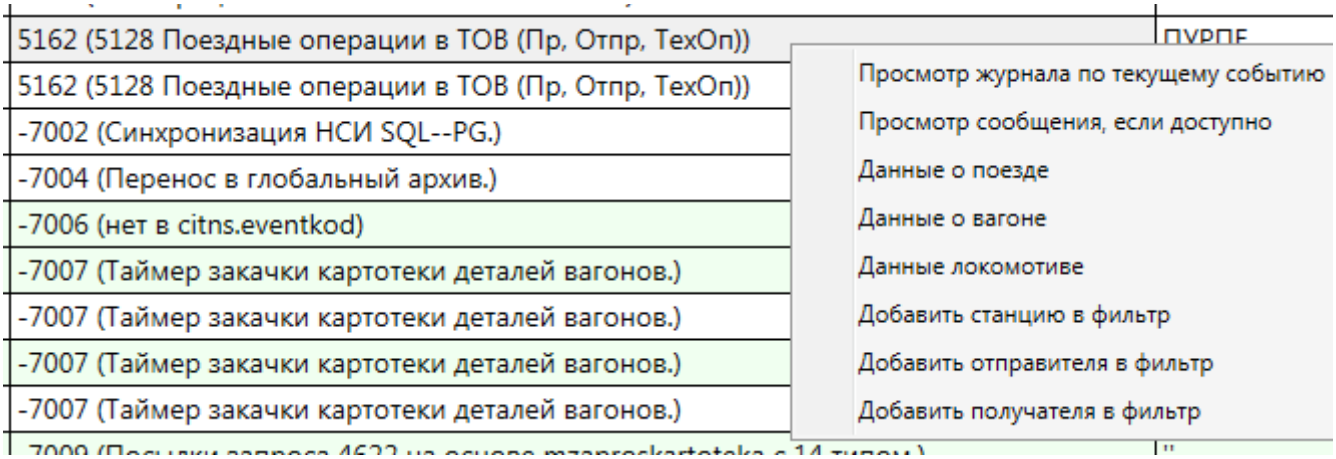


Рисунок 20

При просмотре журнала, введённые фильтры сразу отображаются как условие WHERE для выборки в поле внизу экрана (рисунок 21)

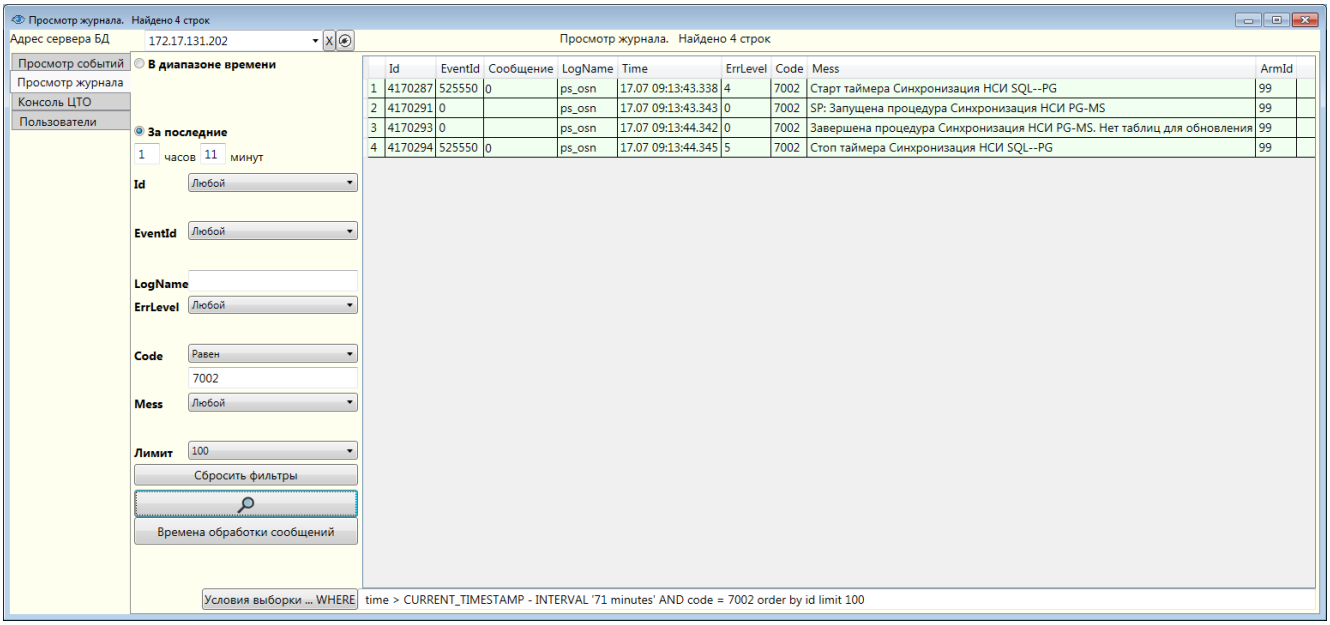


Рисунок 21

Перед нажатием кнопки «найти», текст условия можно скорректировать вручную. Однако, надо помнить, что, при любом изменении условий отбора, ручная корректировка будет потеряна. Двойной щелчок по записи открывает журнал событий +-10 от ID события в этой строке (рисунок 22а, 22б).

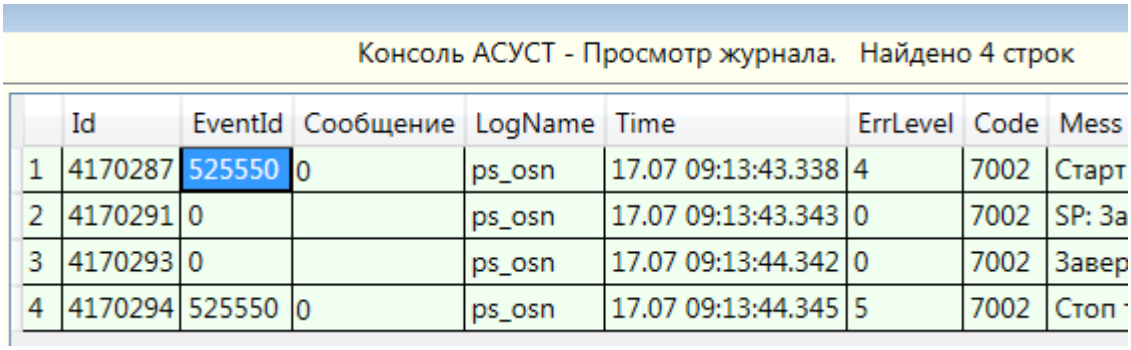
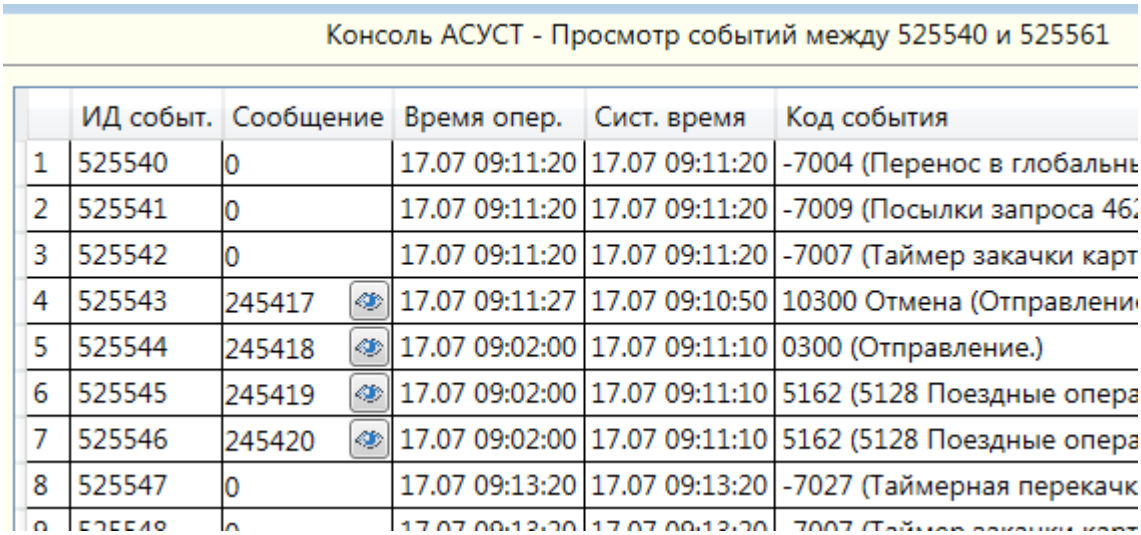


Рисунок 22а



	ИД событ.	Сообщение	Время опер.	Сист. время	Код события
1	525540	0	17.07 09:11:20	17.07 09:11:20	-7004 (Перенос в глобальны
2	525541	0	17.07 09:11:20	17.07 09:11:20	-7009 (Посылки запроса 46
3	525542	0	17.07 09:11:20	17.07 09:11:20	-7007 (Таймер закачки карт
4	525543	245417	17.07 09:11:27	17.07 09:10:50	10300 Отмена (Отправлени
5	525544	245418	17.07 09:02:00	17.07 09:11:10	0300 (Отправление.)
6	525545	245419	17.07 09:02:00	17.07 09:11:10	5162 (5128 Поездные опера
7	525546	245420	17.07 09:02:00	17.07 09:11:10	5162 (5128 Поездные опера
8	525547	0	17.07 09:13:20	17.07 09:13:20	-7027 (Таймерная перекачк

Рисунок 22б

имеется локальное меню (рисунок 23):

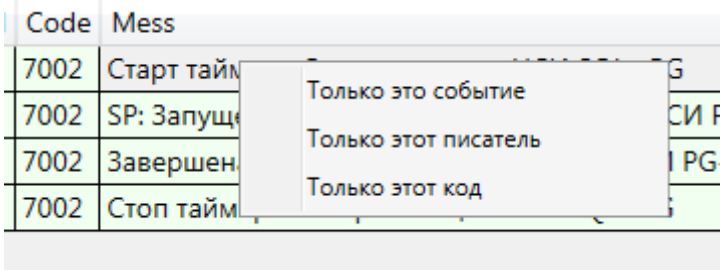


Рисунок 23

Окно Консоли ЦТО (рисунок 24) позволяет увидеть состояние каналов связи ЦТО и очередей в них. Двойной щелчок на канале или выбор пункта локального меню «Просмотр сообщений канала» (рисунок 25)

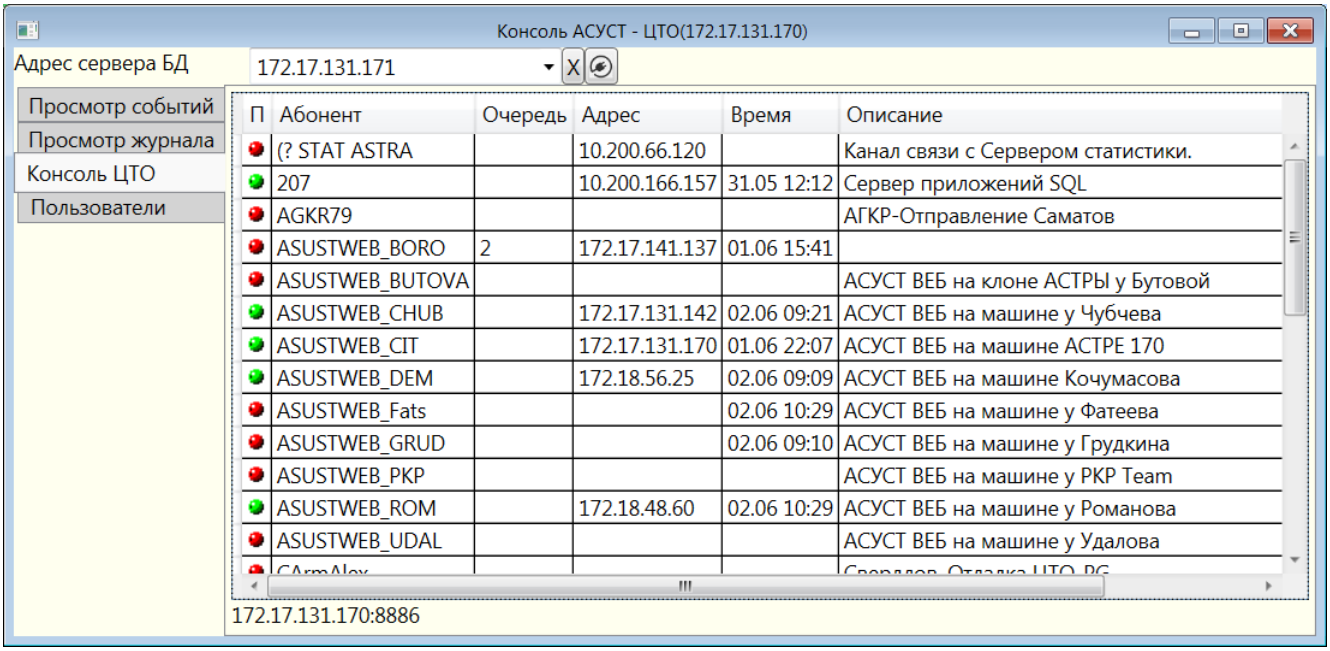


Рисунок 24

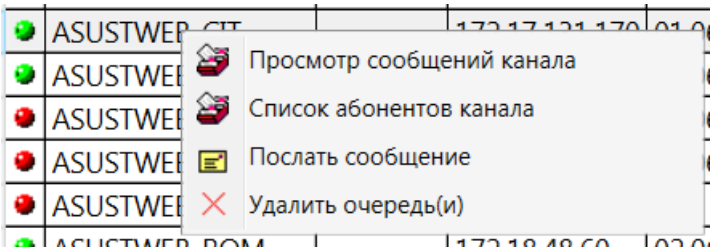


Рисунок 25

открывает окно истории сообщений (рисунок 26).

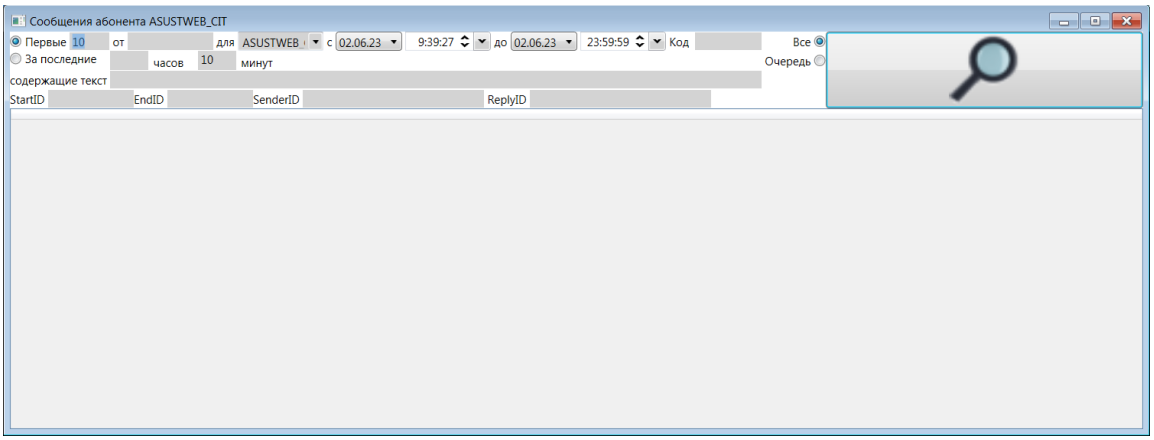


Рисунок 26

Имейте в виду:

1. Выпадающий список дат содержит только даты за которые есть сообщения и сегодня.
2. Код 4097 и 004097 – это разные коды
3. Лимит выдачи – «Первые.»

Окно пользователей, по умолчанию, показывает зарегистрированных в таблице kcmod.UserWork WHERE oper=6 пользователей (рисунок 27а, 27б).

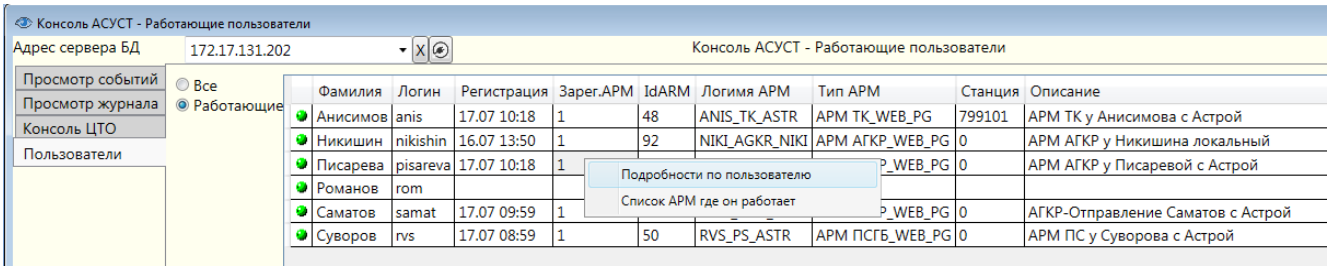


Рисунок 27а

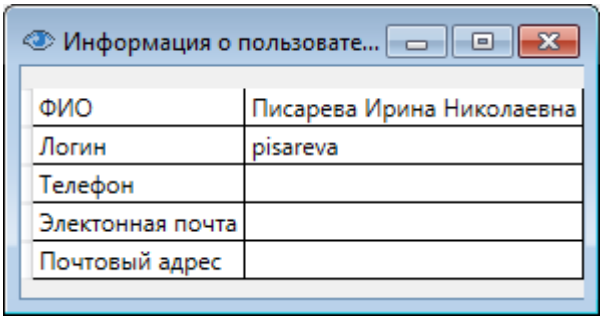


Рисунок 27б

6.2 СРЕДСТВА АДМИНИСТРАТОРА DB POSTGRES

6.2.1 НАСТРОЙКА СЕРВЕРА.

- Команды настройки:

- все настройки в **файле postgresql.conf**,

- alter меняет файл postgresql.auto.conf

- узнать расположение файла настроек:

SELECT * FROM pg_settings **WHERE name LIKE** 'conf%' **ORDER BY name**

- ***show all(имя)*** - показать все(одну) текущие настройки, после последней перезагрузки сервера. Текущие настройки также в представлении *Pg_setting.*

- ***Alter system (set/reset)*** - изменить параметр конфигурации сервера.

- ***select pg_reload_conf()*** - обновить настройки, которые не требуют перезапуска сервера.

```
select * from pg_catalog.pg_settings
show all ;
show shared_buffers ;

alter system set shared_buffers = 512000;          -- блок 8 кб, = 4гб
restart
alter system set work_mem = 65536;                 -- блок 1 кб = 64мб
alter system set effective_cache_size= 1048576;    -- блок 8кб, 8гб,
pg_reload_conf
alter system set maintenance_work_mem= 1048576;    -- блок 1кб, 1гб,
pg_reload_conf
alter system set temp_buffers= 2048;               -- блок 8 кб , 16мб,
pg_reload_conf

alter system reset shared_buffers ;
select pg_reload_conf();
```

sudo su – postgres - смена пользователя на postgres

Перезагрузка:

sudo service postgresql restart

sudo systemctl restart postgrespro-std-12.service - (от имени администратора)

- Параметры настройки:

- *Используемая память*:

- *общий буфер сервера PG:* ***shared_buffers***, данные БД для выполнения активных процессов, для кеширования запросов. Примерно 20-25% доступной памяти. Назначается количество блоков по 8 кб. Требуется перезапуск PG.

- *память для сортировки результата запроса:* ***work_mem***, max ОП, выделяемое одной операции сортировки, агрегации и т.д. Память не разделяемая. Можно выделять 2-4% общей памяти. Для простых запросов меньше, для сложных больше. Например: 1-4Гб – 32-128мб, 32гб- 128мб. Еденица измерения – 1 кб.

- *память для работы задач обслуживания, команды Vacuum:* ***maintenance_work_mem***. ОП, для команд Vacuum, Creat Index и т.д. Размер больше чем размер *work_mem.* Например: 1-4Гб – 128-512мб. Рекомендация RAM/16.

- wal_buffers PostgreSQL сначала записывает записи в WAL (журнал предзаписи) в буферы, а затем эти буферы сбрасываются на диск. Размер буфера по умолчанию, составляет 16 МБ. Но если у вас много одновременных подключений, то более высокое значение может повысить производительность. По умолчанию обычно достаточно.

- effective_cache_size - предоставляет оценку памяти, доступной для кэширования диска. Это всего лишь ориентир, а не точный объем выделенной памяти или кеша. Он не выделяет фактическую память, но сообщает оптимизатору объем кеша, доступный в ядре. Если значение этого параметра установлено слишком низким, планировщик запросов может принять решение не использовать некоторые индексы, даже если они будут полезны. Поэтому установка большого значения всегда имеет смысл.

- *буфер под временные объекты:* ***Temp_buffers***. Примерно 16 мб.

- *специальный стек для сервера:* ***max_stack_depth***. Примерно 2-4 мб.

- Журнал транзакций**:**

- checkpoint_timeout, checkpoint_completion_target

PostgreSQL записывает изменения в WAL. Процесс контрольной точки сбрасывает данные в файлы. Это действие выполняется, когда возникает контрольная точка (CHECKPOINT). Это дорогостоящая операция и может вызвать огромное количество операций IO.

Параметр **checkpoint_timeout** используется для установки времени между контрольными точками WAL. Установка слишком низкого значения уменьшает время восстановления после сбоя, но это также снижает производительность. По умолчанию стоит 5 минут, можно увеличить до 30.

checkpoint_completion_target — это доля времени между контрольными точками для завершения контрольной точки. Высокая частота контрольных точек может повлиять на производительность. Для плавного выполнения задания контрольной точки, **checkpoint_timeout** должен иметь низкое значение. В противном случае ОС будет накапливать все грязные страницы до тех пор, пока соотношение не будет соблюдено, а затем производить большой сброс.

-synchronous_commit

Используется для обеспечения того, что фиксация транзакции будет ожидать записи WAL на диск, прежде чем вернуть клиенту статус успешного завершения. Это компромисс между производительностью и надежностью. Если ваше приложение разработано таким образом, что производительность важнее надежности, отключите **synchronous_commit**. В этом случае транзакция фиксируется очень быстро, потому что она не будет ожидать сброса файла WAL, но надежность будет поставлена под угрозу. В случае сбоя сервера данные могут быть потеряны, даже если клиент получил сообщение об успешном завершении фиксации транзакции.

- Планировщик запросов**:**

- *объем файлового кэша ОС:* ***effective_cashe_size*****,** рекомендуется ставить 60% от общей памяти. Используется для построения плана исполнения запросов.

- Перенос журнала транзакций на отдельный диск**:**

-*перенести* каталоги pg_glog, pgxlog.

6.2.2 ПОЛЕЗНЫЕ КОМАНДЫ DBA

--- **Select version()** -- номер версии сервера.

--- **SELECT pg_size_pretty(pg_database_size(current_database()))**;

- размер БД

--- **SELECT relname, relpages from pg_class ORDER BY relpages DESC LIMIT 10**

- самая большая таблица или индекс.

--- **SELECT * FROM pg_stat_activity** – подключенные пользователи

6.2.2.1 Трассировка

Меняя ряд параметров файла postgresql.conf, можно регулировать количество информации в LOG

alter system set log_statement = 'mod'; -- какие команды (none,all,ddl,mod)

alter system set log_min_duration_statement = 4; -- продолжительность, если поставить 0, то все команды, невзирая на значение в log_statement

select pg_reload_conf();

select **name**,setting,sourcefile **from** pg_settings **where** **name**='log_statement'; - -- в каком конфиге хранится параметр

show log_min_duration_statement

6.2.2.2 Настройка AutoVacuum в PostgreSQL (очистка)

Когда выполняеся DELETE в PostgreSQL, строка не удаляется. Вместо этого устанавливается отметка «удаленное». Также UPDATE в PostgreSQL можно рассматривать как DELETE + INSERT. Это одна из основных идей PostgreSQL, поскольку она обеспечивает больший параллелизм и минимальные блокировки между различными процессами. Недостатком этой реализации является то, что она оставляет удаленные Кортежи, даже после завершения всех транзакций, которые могли видеть эти версии. Если очистка не выполняется, эти «мертвые кортежи» (фактически невидимые для любой транзакции) всегда будут оставаться в файле данных.

Самый простой способ освободить место, занимаемое мертвыми кортежами, - это вручную запустить команду VACUUM, которая просканирует таблицу и удалит мертвые кортежи из таблицы и индекса**. Обычно он не возвращает дисковое пространство операционной системе, но делает его доступным для новых строк.**

Примечание. VACUUM FULL освободит пространство и вернет его операционной системе, но у него много недостатков. Во-первых, он получит эксклюзивную блокировку таблицы, блокируя все операции (включая SELECT). Во-вторых, он фактически создает копию таблицы (Таблица копируется), что удваивает необходимое дисковое пространство, в то время как копирование таблицы происходит очень медленно.

Вышеупомянутые проблемы могут быть устранены по требованию с помощью автоочистки. База данных знает, сколько мертвых кортежей создается с течением времени, поэтому, когда таблица накапливает определенное количество мертвых кортежей, она запускает очистку (по умолчанию это 20% мертвых кортежей) таблицы, но она будет использовать базу данных, и большая часть базы данных имеет меньше времени простоя.

Очистка мертвых кортежей - не единственная задача автоочистки, она также отвечает за обновление статистики распределения данных, используемой оптимизатором при планировании запросов. Вы можете собирать их, вручную запустив ANALYZE, но он сталкивается с аналогичными проблемами VACUUM

Решение аналогично, база данных может отслеживать количество строк, измененных в таблице, и автоматически запускать ANALYZE.

AUTOVACUUM должен быть всегда готов начинать обрабатывать ваши данные, но, желательно, чтобы он не делал этого постоянно – его надо кормить маленькими порциями.

На самом деле autovacuum полностью отключить нельзя — даже при autovacuum = off иногда (после большого количества транзакций) autovacuum будет запускаться.

autovacuum = on

Не забудьте раскомментировать эту строчку в postgresql.conf (убрать впереди знак решетки «#»).

track_counts

Включает сбор статистики активности в базе данных. Этот параметр по умолчанию включён, так как собранная информация требуется демону автоочистки. Не выключать.

autovacuum_naptime

— минимальный интервал, реже которого autovacuum не будет запускаться. По умолчанию 1 минута. И это довольно большое значение. Всегда имеет смысл уменьшать его, мы на практике ставим параметр в 1 секунду. Другие говорят надо увеличить. Остановимся на 20 сек.

autovacuum_max_workers

Сколько фоновых процессов будет вакуумить («пылесосить» вашу базу от “мёртвых” записей)? Причем, надо помнить, что пылесосятся не только таблицы, но и индексы. Индекс в PostgreSQL – это совершенно отдельный файл. Хорошая практика настройки AutoVacuum – это устанавливать количество равным **15-**50% всех ядер сервера СУБД.

autovacuum_vacuum_cost_limi

По умолчанию = -1 (то есть используется значение *vacuum_cost_limit* = 200). Следует учитывать, что предел общий для всех рабочих процессов. При изменении числа одновременно работающих рабочих процессов общая нагрузка будет оставаться постоянной. Поэтому, если надо увеличить производительность автоочистки, то при добавлении рабочих процессов стоит увеличить и *autovacuum_vacuum_cost_limit*.

autovacuum_vacuum_cost_delay

По умолчанию= 20ms. На современной аппаратуре с этими цифрами автоочистка будет работать очень и очень медленно. В версии 12 значение *autovacuum_vacuum_cost_delay* будет уменьшено до 2ms, что можно считать более подходящим первым приближением.

autovacuum_vacuum_scale_factor

По умолчанию, настройка при которой AutoVacuum сработает на таблице, равна 0.1 или 10% (т.е. с последнего AutoVacuum в этой таблице должно поменяться 10% строк). Надо ее поменять хотя бы до 5 процентов

autovacuum_analyze_scale_factor

А autovacuum_analyze_scale_factor по умолчанию вообще равен 0.2 или 20%.

То есть, PostgreSQL статистику по умолчанию посчитает только после изменения 20%. Потому что в нормальных приложениях и в нормальных базах данных люди практически всегда пользуются всей таблицей – там считается, что архивные данные, которые никому не нужны, должны лежать в соседнем месте. Как настраивать? Параметр autovacuum_analyze_scale_factor ужесточаем в 10-20 раз, до 1- 2%.

autovacuum_vacuum_threshold, autovacuum_analyze_threshold

— количество измененных или удаленных записей в таблице, необходимых для запуска процесса сборки мусора VACUUM или сбора статистики ANALYZE. По умолчанию по 50. Порог в 50 строк предназначен для предотвращения очень частой очистки крошечных таблиц

Можно- игнорировать масштабный коэффициент в postgresql.conf и установить больший порог (например, установить autovacuum_vacuum_scale_factor = 0 и autovacuum_vacuum_threshold = 10000), а затем в соответствии с частотой удаления и обновления каждой таблицы и объемом данных в таблице отдельно Установите порог для каждой таблицы:

--ALTER TABLE t SET (autovacuum_vacuum_scale_factor = 0);

ALTER TABLE t SET (autovacuum_vacuum_threshold = 100)

6.2.3 РЕЗЕРВНОЕ КОПИРОВАНИЕ И ВОССТАНОВЛЕНИЕ

1. Простейший вариант:

su - postgres

Сделайте дамп содержимого базы данных в файл, выполнив следующую команду.

pg_dump dbname > dbname.bak

База данных, заданная параметром *имя_базы*, не будет создана данной командой, так что вы должны создать её сами перед запуском psql.

dropdb dbname

createdb dbname

Восстановите базу данных, используя psql:

psql test < dbname.bak

2. Утилиты

pg_dump — подходит для случаев когда нужно сделать резервную копию таблицы, базы, схемы или данных. *pg_ dumpall* — подходит для случаев когда нужно сделать резервную копию целиком всего кластера БД

pg_restore – восстановление дамп, выгруженного не в текстовом формате

psql – для чтенич текстовых файлов(скриптов), созданные pg_dump,. Общий вид команды для восстановления дампа: psql *имя_базы* < *файл_дампа*

6.3 МОНИТОРИНГ ПРОБЛЕМ ПРОИЗВОДИТЕЛЬНОСТИ

- ***PG_STAT_STATEMENTS***

Чтобы включить pg_stat_statements на вашем сервере, измените следующую строку в postgresql.conf и перезапустите PostgreSQL:

shared_preload_libraries = ‘pg_stat_statements’

ALTER SYSTEM SET shared_preload_libraries = pg_stat_statements;

После загрузки этого модуля на сервер PostgreSQL автоматически начнет собирать информацию. Хорошо то, что накладные расходы модуля действительно очень низкие (накладные расходы в основном просто «шум»).

Затем выполните следующую команду для создания представления, необходимого для доступа к данным:

CREATE EXTENSION pg_stat_statements;

Расширение создаст представление с именем pg_stat_statements и сделает данные легко доступными. Схема Public База текущая.

Время отклика: SELECT sum(total_time)/SUM(calls) **FROM pg_stat_statements**

Интересные запросы — отсортировать вывод pg_stat_statements по total_time:

SELECT * FROM pg_stat_statements ORDER BY total_time DESC;

Следующий запрос **показывает 20 запросов, занимающих много времени**:

```
SELECT substring(query, 1, 50) AS short_query,
        round(total_time::numeric, 2) AS total_time,
        calls,
        round(mean_time::numeric, 2) AS mean,
        round((100 * total_time / sum(total_time::numeric) OVER ())::numeric, 2) AS percentage_cpu
FROM  pg_stat_statements
ORDER BY total_time DESC
LIMIT 20;
```

- ***AUTO_EXPLAIN***

Для автоматического логирования планов выполнения существует модуль auto_explain

Модуль auto_explain предоставляет возможность автоматического протоколирования планов выполнения медленных операторов, что позволяет обойтись без выполнения EXPLAIN вручную. Это особенно полезно для выявления неоптимизированных запросов в больших приложениях.

Т.е. в Postgres возможно узнать план запроса на момент его исполнения

Сделать это можно двумя путями:

1. Включив модуль для запроса:

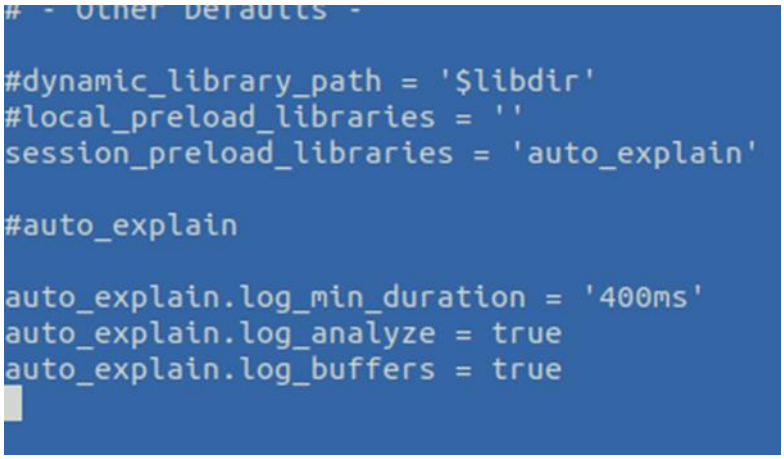
load 'auto_explain';

SET auto_explain.log_min_duration = 10;

SET auto_explain.log_analyze = true;

Это полезно, если попрофилировать нужно конкретный запрос

Включить модуль на все запросы на сервере



Не забудь применить конфиги

SELECT pg_reload_conf();

■ *POSTGRES STATISTICS COLLECTOR (ТЮНИНГ ИНДЕКСОВ)*

Сборщик статистики в PostgreSQL представляет собой подсистему, которая собирает и отображает информацию о работе сервера. В настоящее время сборщик может подсчитывать количество обращений к таблицам и индексам — в виде количества прочитанных блоков или строк с диска.

Кроме того, он отслеживает общее число строк в каждой таблице, информацию о выполнении очистки и сбора статистики для каждой таблицы. Он также может подсчитывать вызовы пользовательских функций и общее время, затраченное на выполнение каждой из них.

Поскольку сбор статистики несколько увеличивает накладные расходы при выполнении запроса, есть возможность настроить СУБД так, чтобы выполнять или не выполнять сбор статистической информации. Это контролируется конфигурационными параметрами, которые обычно устанавливаются в файле postgresql.conf.

Параметр track_activities включает мониторинг текущих команд, выполняемой любым серверным процессом.

Параметр track_counts определяет необходимость сбора статистики по обращениям к таблицам и индексам.

Параметр track_functions включает отслеживание использования пользовательских функций.

■ *ОТСУТСТВУЮЩИЕ ИНДЕКСЫ*

Отсутствующие индексы может быть одним из самых простых решений для повышения производительности запросов. Если у вас включен сборщик статистики, вы можете выполнить следующий запрос (источник).

```
SELECT
  relname,  seq_scan – idx_scan AS too_much_seq,
  CASE
    WHEN       seq_scan – coalesce(idx_scan, 0) > 0
    THEN       ‘Missing Index?’
    ELSE       ‘OK’
  END,
  pg_relation_size(relname::regclass) AS rel_size, seq_scan, idx_scan
FROM   pg_stat_all_tables
```

```
WHERE   schemaname = ‘public’   AND pg_relation_size(relname::regclass)
> 80000
ORDER BY   too_much_seq DESC;
```

Запрос находит таблицы, в которых было больше последовательных сканирований (Sequential Scans), чем индексных сканирований (Index Scans) — явный признак того, что индекс поможет.

- ***НЕИСПОЛЬЗУЕМЫЕ ИНДЕКСЫ***

Чтобы найти неиспользуемые индексы, вы можете выполнить следующий запрос.

```
SELECT
  indexrelid::regclass as index,  relid::regclass as table,
  ‘DROP INDEX ‘ || indexrelid::regclass || ‘;’ as drop_statement
FROM   pg_stat_user_indexes   JOIN
    pg_index USING (indexrelid)
WHERE   idx_scan = 0   AND indisunique is false;
```

6.4 ПОДСИСТЕМА ЗАПУСКА ТАЙМЕРНЫХ ЗАДАЧ(JOB)

В связи с тем, что в Postgres нет механизма запуска таймерных задач, в планировщике создан механизм, который может запускать по расписанию SQL команды. Для запуска таких(SQL) команд, необходимо прописать их в таблицах описания таймеров, выставив признак iscommand = 1.

После описания нового таймера, необходимо перезагрузить планировщик, для обновления библиотек.

6.4.1 ОПИСАНИЕ НСИ И ПРИНЦИПОВ РАБОТЫ ТАЙМЕРОВ

Создана таблица НСИ, в которой должны быть описаны все JOB:

```
CREATE TABLE citns.srvtimers (
  timerid int4 NOT NULL,--идентификатор таймера
  timertype int2 NOT NULL DEFAULT 0,--тип таймера
  timeday int4 NOT NULL DEFAULT 0, --в зависимости от типа
  timehour int2 NOT NULL, --в зависимости от типа
  timeminute int2 NOT NULL, --в зависимости от типа
  maxtime int4 NOT NULL DEFAULT 0,--максимальное время работы( в чем)
  description varchar NULL,--описание
  iscommand int2 NOT NULL , 1 -выполняется заданная SQL команда.
  commandtext text NULL, --выполняемая команда
  CONSTRAINT pk_nssrvtimers PRIMARY KEY (timerid)
);
```

TimerId - для чисток от 100 000, для других – свой уникальный тип таймера:

0 - Интервальный,

1 – По дням недели,

2 - Месячный

Интервальный – Частота выполнения таймера задается полями timehour, timeminute.

По дням недели – Дни недели запуска задаются битами в поле timeday. Время запуска в полях timehour, timeminute.

Месячный - Дни месяца запуска задаются битами в поле timeday. Время запуска в полях timehour, timeminute.

Задание само отвечает, за то, чтобы отрабатывать только на нужном типе сервера (АСУ ЖДП,ССП,ТК….)

Все задания, работу которых надо временно отключить или увеличить лимит времени, должны быть описаны в таблице citnv.srvtimers:

```
CREATE TABLE citnv.srvtimers (
  timerid int4 NOT NULL,
  esvkl bool NOT NULL DEFAULT 0::boolean,-- 0- отключен, 1 - включен
  maxtime int4 NULL,
  CONSTRAINT pk_nvsrvtimers PRIMARY KEY (timerid)
);
```

6.4.2 НАБЛЮДЕНИЕ ЗА РАБОТОЙ ЗАДАНИЙ

Предлагается сделать задания полноценным участником технологического процесса АСУ ЖДП. Т.е. предлагается каждому заданию иметь свой технологический код, по аналогии с кодом сообщения. Причем коды сообщений и коды заданий должны иметь сквозную нумерацию (диапазон 7000-7500).

Для ведения модели создаем таблицу Kcmod.StatTimers.

```
CREATE TABLE kcmod.stattimers (
timerid int4 NOT NULL, - ИД таймера
laststarttime timestamp NULL, - Время последнего запуска таймера
lastendtime timestamp NULL, - Время последнего окончания таймера
isfault int2 NULL, - статус (0 – успешно завершен, 1- завершен с ошибкой, 2- работает) статус меняется раз в минуту.
Lastresult varchar NULL- текст ошибки
);
```

```
CREATE UNIQUE INDEX ix_stattimers_timerid ON kcmod.stattimers USING btree (timerid);
```

Для ведения истории выполнения заданий создается таблица Kcmod.StatTimersArx.

```
CREATE TABLE kcmod.stattimersarx (
timerid int4 NOT NULL, - ИД таймера
 starttime timestamp NULL, - Время запуска таймера
worktime int4 NULL, - Продолжительность работы
isfault int2 NULL, - признак завершения (0- успешно, 1- ошибка)
 lastresult varchar NULL- Текст ошибки
);
CREATE INDEX ix_stattimersarx_timerid ON kcmod.stattimersarx USING btree (timerid);
```

Для выявления одновременно работающих заданий, используем просмотр kcmod.vwstattimersarx. Он показывает не все варианты одновременной работы, но для оценки более чем достаточно. Для текущего состояния kcmod.vwstattimers.

Начало и конец работы таймера также фиксируется в таблице kcmod.journal, с кодами 4 и 5.

Появилась возможность менять время запуска и (только для таймеров, выполняющих команды SQL) текст команды на ходу, без перезагрузки ПС. Включение- отключение таймеров без перезагрузки не возможно.

6.5 ПРАВИЛА ИМЕНОВАНИЯ ОБЪЕКТОВ БД

1. При создании нового объекта базы (процедуры, таблицы, просмотра), необходимо создавать **Comment**, в котором следует указать

- автора,

- дату последней корректировки,

- назначение объекта.

При дальнейшей работе с объектом следить за сохранностью и актуальностью Commenta.

2. Имя новой процедуры или функции формируется следующим образом:

- первый символ (p,f,t) – процедура, функция, триггер

- 2-4 символы – тема- **обязательно** (tov, dsp, sks…….)

- 5-6 символ – автор- **желательно** (CG – Чубчев Геннадий)

- далее осмысленное название процедуры (InsertVagon)

Пример: fSksCgGetFreeObjID.

3. Перечень тем:

Ac – Отчетность

Cnt - Контейнера

Dsp – ДСП

Sks – Сервер АСУ ЖДП

Spr – Справочная система

Tk – АРМ ТК

TO – Телеобработка

Tov - ТОВ

Twk – Грузовая работа

Tmp- Неопределено

4. Список авторов:

CG - Чубчев

KG - Карнаухова

MI - Максимов

MS - Морева

SA Свердлов

TA - Томин

5. При создании новых таблиц все поля по умолчанию NOT NULL, кроме случаев когда это точно необходимо. Не использовать для полей DEFAULT. Менять данные в таблицах через процедуры.

6.6 ПРИНЦИПЫ ОБМЕНА СООБЩЕНИЯМИ С ТОНКИМ КЛИЕНТОМ

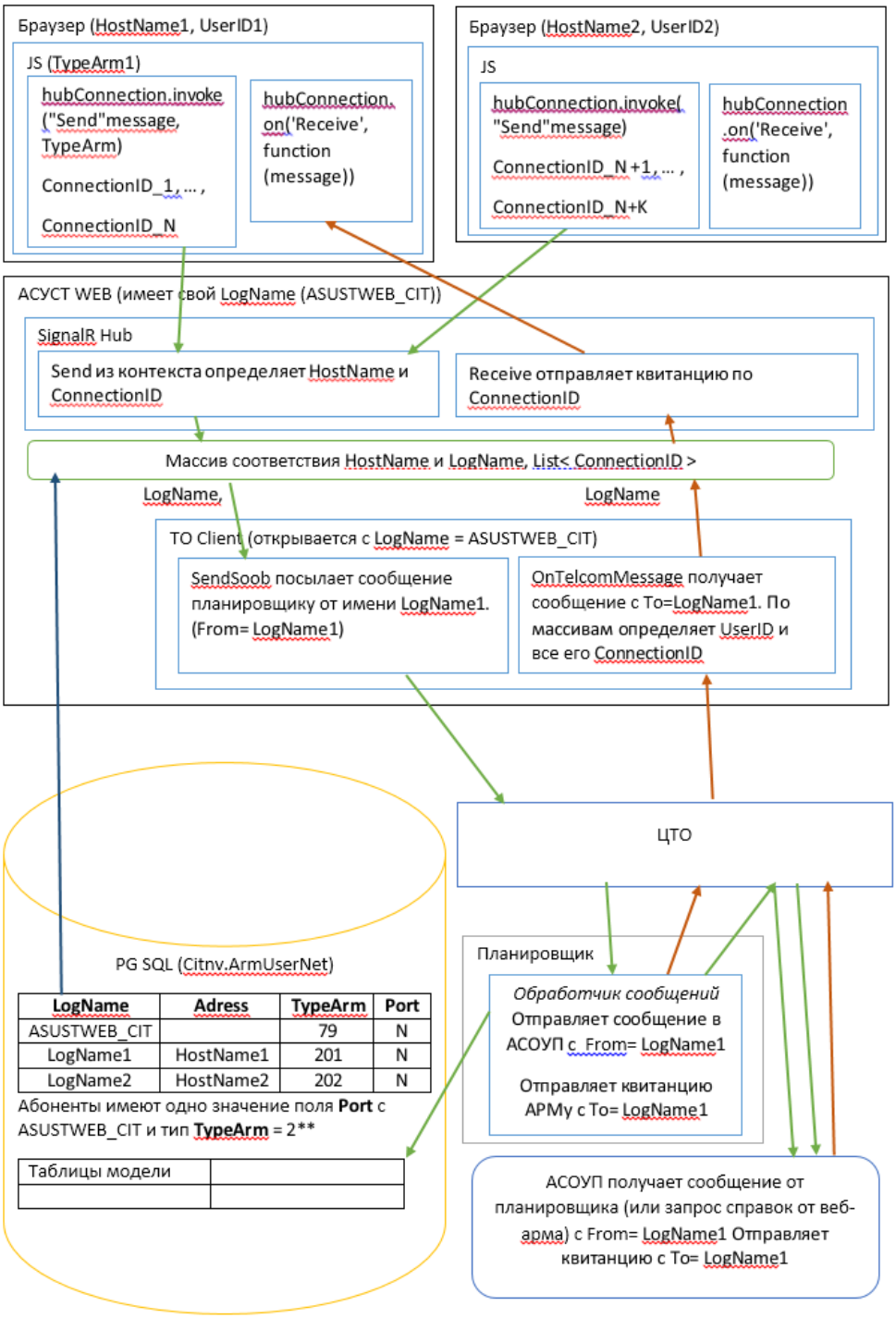


Схема 1

6.6.1 ТЕРМИНЫ

1. **Пользователь** – человек работающий на компьютере.

2. **Юзер** – понятие используемое при авторизации (логин/пароль). Юзер имеет свои полномочия (Роли, операции, инфраобъекты). В обмене

сообщениями не имеет технического значения. Используется как составляющая сообщения 4151 о начале и завершении работы.

3. **Имя компьютера** (далее HostName) – сетевое имя компьютера (HostName) клиента. Используется для идентификации рабочего места. В условиях повсеместного использования динамических IP-адресов, использование для идентификации именно HostName наиболее оправдано.

4. **Лог-имя** (далее LogName) – идентификатор абонента в системе телеобработки АСУ ЖДП.

5. **Тип АРМа** – определяется технологическим типом АРМ и его программной реализацией (Win ТК, Web ТК). В таблице kcmod..nsArmType это поле ArmType_id. В таблице Citnv..ArmUserNet это TypeArm . За АСУ ЖДП WEB закреплён тип АРМа 79. У всех веб-АРМов тип лежит в диапазоне 200-255

6. **АСУ ЖДП WEB** – комплекс сборок, устанавливаемых на сервере, обеспечивающий функционал работы с тонким клиентом. АСУ ЖДП WEB работает под управлением WEB-сервера Nginx. Состоит из оболочки и компонентов.

7. **Компонент –** комплекс сборок в составе АСУ ЖДП WEB обеспечивающий функционал работы одного АРМа в старом понимании (АРМ ТК, АРМ ТОВ, Справочная система и т.д.)

8. **Оболочка –** комплекс сборок в составе АСУ ЖДП WEB обеспечивающий координацию, и общие функции Веб-АРМов, такие как авторизация, загрузка НСИ, печать, логгирование, сохранение пользовательских настроек, стиль элементов управления и др.

9. **Страница –** видимый интерфейс в окне браузера, обеспечивающий взаимодействие с пользователем (клиентский комп), и выполняющий одну технологическую задачу.

10. **Веб-АРМ –** комплекс страниц, обеспечивающий функционал работы одного АРМа в старом понимании (АРМ ТК, АРМ ТОВ, Справочная система

и т.д.) В технологическом смысле веб-АРМ ассоциируется с типом АРМ (TypeArm)

11. **Классический АРМ –** приложение-толстый клиент (АРМ ТК, АРМ ТОВ, Справочная система и т.д.)

12. **Скрипт –** программный модуль на странице (клиентский комп), обеспечивающий взаимодействие пользователя с АСУ ЖДП WEB.

13. **Планировщик –** приложение, устанавливаемое на сервере, обеспечивающие в том числе обработку сообщений от АСУ ЖДП WEB, запись данных в базу, посылку сообщений в АСОУП, рассылку сообщений на обновление страницам Веб-АРМов через АСУ ЖДП WEB

6.6.2 ОСНОВНЫЕ ПОЛОЖЕНИЯ

1. Пользователь под одним юзером может запустить несколько АРМов с разными типами на одном компе.

2. Пользователь под одним юзером может запустить несколько экземпляров веб-АРМов на одном компе.

3. Пользователь под одним юзером может запустить только один экземпляр классического АРМа одного типа на одном компе.

4. Пользователь под одним юзером может запустить несколько АРМов одного типа на разных компах.

5. Внутри АСУ ЖДП WEB пользователь, согласно полномочий юзера, имеет доступ к различным Веб-АРМам.

6. При подписке на обновление, которая осуществляется посылкой отдельного сообщения о котором будет написано в отдельном документе веб-АРМ может зарегистрироваться только за одну станцию. При выборе другой станции ему нужно разрегистрироваться и заново зарегистрироваться за новую станцию.

7. АСУ ЖДП WEB читает информацию из базы данных PG SQL. Для записи в базу АСУ ЖДП WEB посылает сообщение Планировщику. При необходимости Планировщик посылает сообщение в АСОУП от имени Веб АРМ. Веб-АРМ напрямую не посылает сообщение оперативной работы в АСОУП, но может напрямую запрашивать справки из АСОУП.

8. Механизм рассылки сообщений на обновление состояния в данном документе пока не рассматривается.

6.6.3 ОБМЕН СООБЩЕНИЯМИ

6.6.3.1 Прямое сообщение (Страница > Планировщик)

6.6.3.1.1 Взаимодействие страниц и АСУ ЖДП WEB.

В данном разделе используются термины *клиент* – Страница и *сервер* - АСУ ЖДП WEB

Клиент и сервер взаимодействуют между собой, используя технологию SignalR, которая (в двух словах) подразумевает создание на стороне сервера хаба (Hub) на прослушивание (Send) и посылку (Receive) сообщений. На стороне клиента в скрипте (JS) формируются блоки подключения к хабу (на схеме не показан), посылки (Invoke(“Send”)) и получения (.on('Receive') сообщения

Пользователь авторизуется в системе под своим юзером и получает доступ к странице веб-АРМа, на которой, допустим, есть кнопка «Послать сообщение» Пользователь нажимает кнопку, страница формирует сообщение и посылает его на сервер.

При запуске АСУ ЖДП WEB инициализируется массив экземпляров класса, содержащего свойства HostName, LogName и List<ConnectionID>. Назовём его **ConnectionList**.

При подключении клиента к серверу в хабе из контекста соединения определяется ConnectionID (идентификатор соединения. Меняется при каждом обновлении страницы, поэтому необходимо иметь массив ConnectionID) и HostName.

Если в ConnectionList уже есть элемент с таким HostName, к нему добавляется ConnectionID.

Если нет, делается запрос в БД таблицу Citnv.ArmUserNet, где по Adress = HostName и TypeArm/100 =2 определяется LogName. Новый элемент с HostName1, LogName1 и ConnectionID1 заносится в массив ConnectionList. При этом из контекста определяется юзер, и планировщику посылается сообщение 4151 о том что UserID1 с LogName1 начал работу (для записи в ArmWork)

Удаление элемента из ConnectionList, происходит по событию отключения последнего ConnectionID из элемента. При этом из контекста определяется юзер, и планировщику посылается сообщение 4151 о том что UserID1 с LogName1 закончил работу (для записи в ArmWork)

6.6.3.2 Взаимодействие АСУ ЖДП WEB и ЦТО

АСУ ЖДП WEB имеет своё лог-имя в таблице Citnv.ArmUserNet (пусть это будет для примера ASUSTWEB_CIT) и тип АРМа 79 У всех веб-АРМов тип лежит в диапазоне 200-255. Инициализация и старт клиента телеобработки осуществляется с лог-именем ASUSTWEB_CIT

Сообщение посылается в адрес Планировщика с параметром From = LogName1, то есть от лог-имени рабочего места а не АСУ ЖДП WEB. Важно, чтобы при прописывании лог-имён (LogName1) в Citnv.ArmUserNet, группа ТО (поле Port) АРМа совпадала с группой ASUSTWEB_CIT (назовём её *группа N*). Тогда, взаимодействовать с ЦТО будет один абонент (ASUSTWEB_CIT) от имени другого (LogName1)

6.6.3.3 Действия Планировщика

Планировщик, получая сообщение от ЦТО отдаёт его обработчику сообщений в котором происходит, запись в базу, если необходимо, отправка сообщения в АСОУП

6.6.3.4 Взаимодействие Планировщика и АСОУП

Планировщик посылает сообщение АСОУП от имени LogName1 (From= LogName1)

6.6.3.5 Обратное сообщение (Планировщик (АСОУП) > Страница)

6.6.3.5.1 Взаимодействие Планировщика (АСОУП) и ЦТО.

Обратное сообщение (квитанцию, сообщение на обновление) Планировщик и АСОУП посылает на имя LogName1 (To= LogName1)

6.6.3.5.2 Взаимодействие ЦТО и АСУ ЖДП WEB

Поскольку подключение к ЦТО осуществляет только один абонент с группой N (см. п. 1.2.3) то все сообщения этой группы приходят на лог-имя ASUSTWEB_CIT, то есть в АСУ ЖДП WEB (модуль OnTelcomMessage)

6.6.3.5.3 Взаимодействие АСУ ЖДП WEB и страниц

Из модуля OnTelcomMessage по LogName1 в массиве ConnectionList (см. п. 1.1.3) находится список ConnectionID

По всем ConnectionID рассылается сообщение странице (hubConnection.on('Receive'…)), где обрабатывается соответствующим образом.

6.7 ОПИСАНИЕ КОНСОЛИ ПЛАНИРОВЩИКА

Консоль планировщика служит для управления и визуализации основных моментов работы службы. Окно консоли имеет вид (рисунок 28):

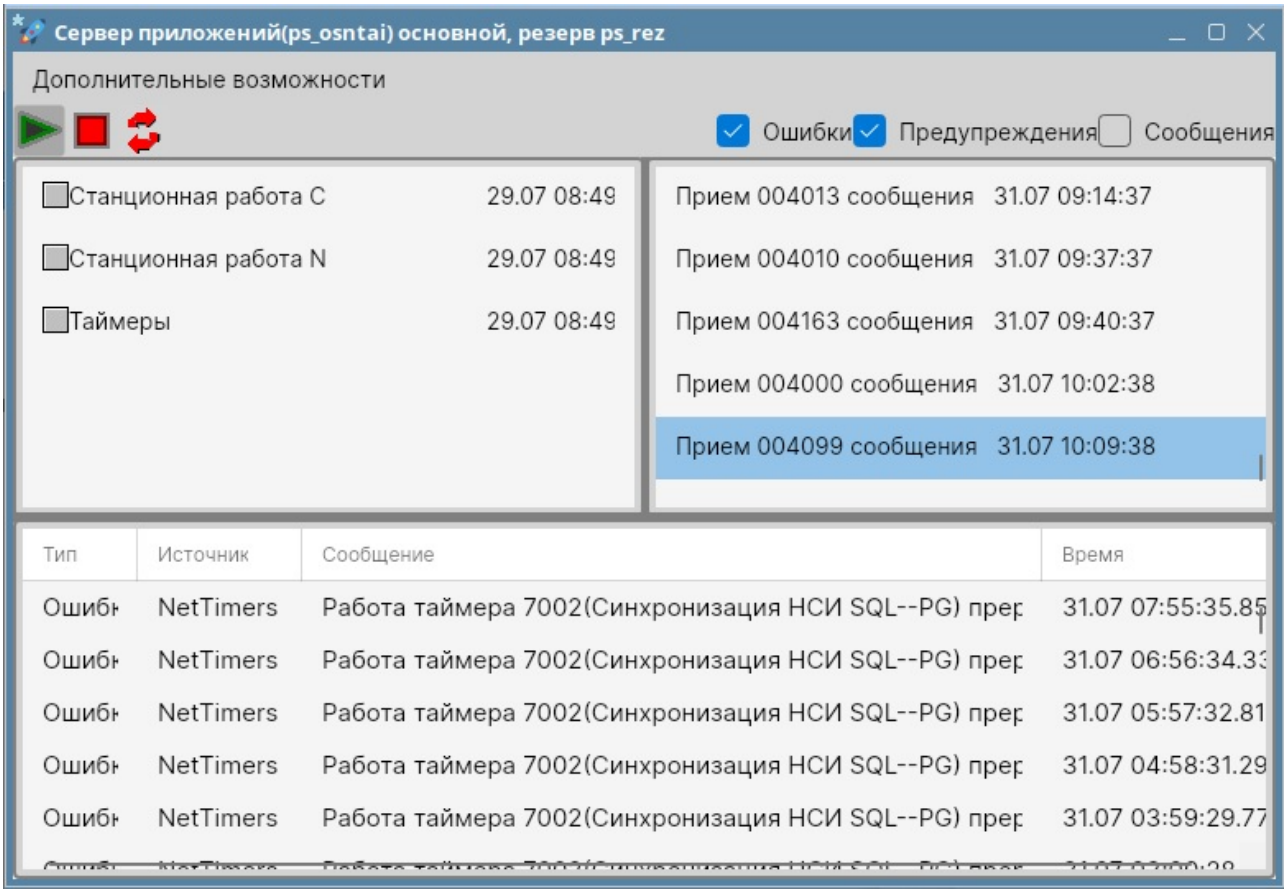


Рисунок 28

И состоит из нескольких частей.

Меню (рисунок 29)

Рисунок 29

Слева кнопки управления службой: запуск, остановка, переподключение к службе.

Справа параметры вывода на экран журнала работы службы

Окно загруженных компонент (рисунок 30)

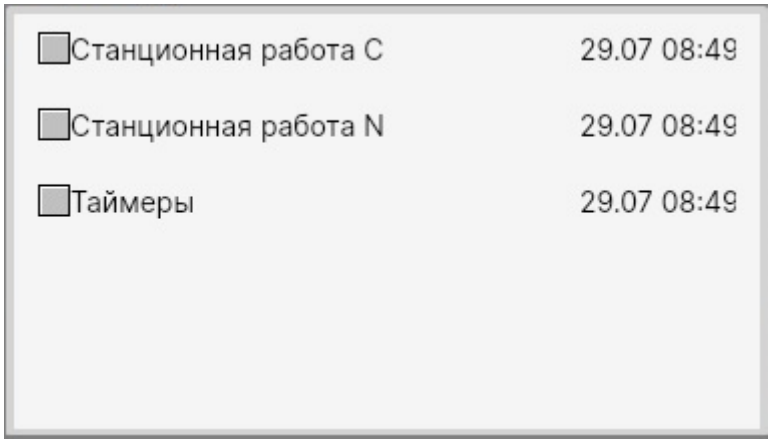


Рисунок 30

Выдает загруженные компоненты и время последнего старта. Если кликнуть правой кнопкой мыши на конкретный компонент, откроется более детальная информация (рисунок 31)

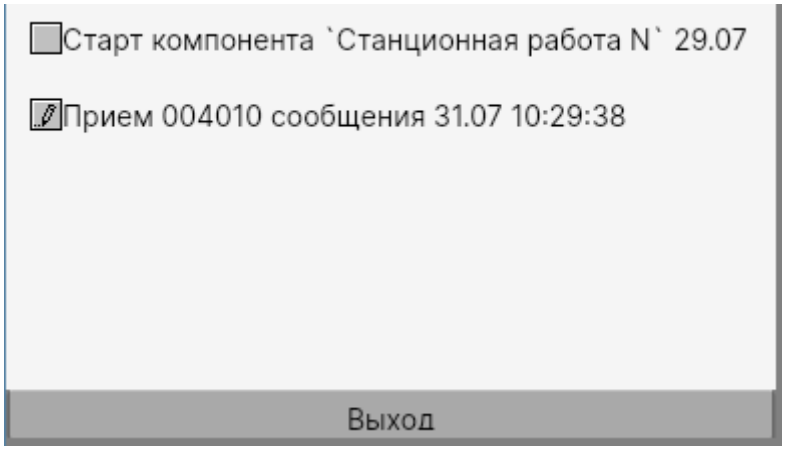


Рисунок 31

Окно последних принятых сообщений. Ведется в реальном времени (рисунок 32)

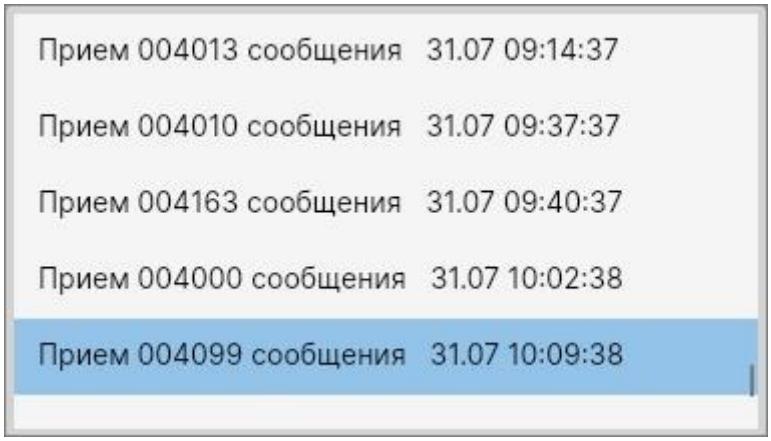


Рисунок 32

Ведется в реальном времени

Окно журнала работы службы (рисунок 33)

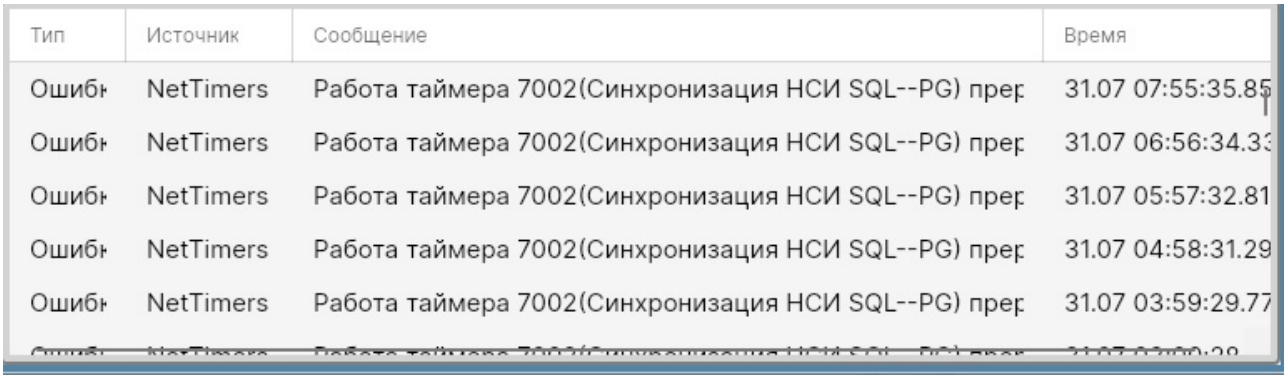


Рисунок 33

Обновляется раз в 5 минут. По правой кнопке мыши можно просмотреть более детальную информацию, если она есть.

6.8 ВОЗМОЖНЫЕ ОШИБКИ ПРИ ЭКСПЛУАТАЦИИ ПЛАНИРОВЩИКА

Так как служба не имеет видимого интерфейса, все ошибки и сообщения записываются в системный журнал событий. Для удобства пользователя, все эти события, за 3 последних дня, выводятся в окне консоли СП.

-Создайте каталог SERVSOOB.XXX(где XXX – логномер СП)

-Вы либо не создали этот каталог, либо неправильно прописали логномер СП в SpService.ini.

-Не удалось загрузить все программы обработки сообщений.

-На консоли, в окне «КОМПОНЕНТЫ», у программы обработки сообщений, которую не смогли загрузить, будет стоять «!». Необходимо просмотреть ошибки в журнале работы службы.

-У ВАС ОТСУТСТВУЕТ ЛИЦЕНЗИЯ ЦИТ_ТРАНС НА ЗАПУСК СП КСАРМ

-Проверьте наличие citini.cit в каталоге CITTRANS\SP_KSARM\SERVSOOB.XXX(где XXX логномер СП)

Если менялся IP сервера – дату этого файла.

-Не смог определить свой логномер.

Надо прописать логномер СП в таблице nvIP для основного сервера Type_serv = 1, Vid_serv = 0, is_DB = 0 и Type_serv = 1, Vid_serv = 1, is_DB = 0 для резервного.

У Вас не прописан IP SQL Server центральной телеобработки в таблице nvIP.

Прописать IP адрес SQL сервера, с которым работает ЦТО (где хранятся очереди ЦТО) в nvIP где Type_serv = 2, Vid_serv = 0, is_DB = 1

Проверьте загружена ли центральная ТО!

ТО СП не смогла соединиться с ЦТО. Либо ЦТО не загружена, либо неправильно прописан адрес в nvIP. Если все загружено и прописано, необходимо посмотреть в telecom.log причину отсутствия связи.

Ошибка инициализации телеобработки TCP/IP\r\n Может центральная ТО не загружена?

ТО СП не смогла соединиться с ЦТО. Либо ЦТО не загружена, либо неправильно прописан адрес в nvIP. Если все загружено и прописано, необходимо посмотреть в telecom.log причину отсутствия связи.

В консоли СП, в окне «КОМПОНЕНТЫ», у программ обработки сообщений, стоит не дата загрузки, а «не загружен».

ТО СП не смогла соединиться с ЦТО. Либо ЦТО не загружена, либо неправильно прописан адрес в nvIP. Если все загружено и прописано, необходимо посмотреть в telecom.log причину отсутствия связи.

Ошибка соединения со службой СП

Возможно была попытка соединиться со службой с другого компьютера либо служба автоматически перегрузилась. Восстановить соединение модно нажав на кнопку с двумя стрелками «Подключиться к службе» на панели инструментов

Рестарт компонента 'Станционная работа' при 1042

Сбои в обработке сообщения.

При загрузке контролируйте наличие записей «Старт службы СП» «Стоп службы СП»

Если Вы загружаете СП, а записи «Старт службы СП» с текущим временем не появляется, возможно служба не смогла остановиться в момент последней остановки. Тогда снимите задачу

Spservece.out через системный монитор.

6.9 РЕЗЕРВНОЕ КОПИРОВАНИЕ

Логическая резервная копия – это набор команд SQL восстанавливающий кластер (БД или отдельный объект) с нуля.

pg_dump – утилита для создания создания резервных копий PostgreSQL. Она позволяет создавать целостные резервные копии отдельных баз данных, не препятствуя работе с базой данных.

Создание резервной копии локальной базы данных (параметр –U не обязательный):

pg_dump –U username dbname > /tmp/dbname.sql

Создание резервной копии отдельной таблицы локальной базы данных:

pg_dump -t example_table dbname > /tmp/dbname_example_table.sql

pg_dumpall – это утилита для создания резервных копий PostgreSQL, позволяет создать резервную копию всех баз данных внутри одного инстанса PostgreSQL, также выгружает глобальные объекты, общие для всех баз данных, такие как роли и табличные пространства.

Создание резервной копии локальной базы данных (параметр –U не обязательный):

pg_dumpall > -U username /tmp/dump.sql

Результатом работы утилит pg_dump и pg_dumpall является скрипт для sql, для восстановления резервной копии можно воспользоваться консольной утилитой psql. Пример использования утилиты psql:

psql –d dbname < /tmp/dbname.sql