Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Национальный исследовательский ядерный университет «МИФИ»

УДК 004.415.2

ОТЧЁТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

РАЗРАБОТКА ПОДСИСТЕМЫ УПРАВЛЕНИЯ ДАННЫМИ КОМПЛЕКСА ПРОМЕЖУТОЧНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ SPD ONLINE FILTER

Студент _____ П. А. Коршунова

Научный руководитель _____ Д. А. Олейник

Москва 2025

Содержание

Введение

Одной из неразрешенных проблем современной физики высоких энергий является «спиновый кризис», который заключается в том, что мы не знаем как спины нуклонов распределены между их составляющими (кварками и глюонами).

Данный кризис был вызван экспериментом EMC [1], в котором пытались определить распределение спина внутри протона. Ожидалось, что весь спин протона несут валентные кварки, однако оказалось, что это не так.

Изучение спиновой структуры нуклона имеет большое значение, так как он отвечает за фундаментальные свойства природы. Но наши знания о его внутренней структуре все еще ограничены, особенно в отношении вклада глюонов. Именно поэтому строится новая установка SPD (являющаяся часть ускорительного комплекса NICA) для всестороннего изучения глюонного состава нуклона [2].

Конструктивной особенностью детектора SPD является отсутствие «классической» триггерной системы, позволяющей реализовать отбор собираемых для дальнейшего исследования событий. Это приводит к необходимости собирать с подсистем весь набор произведенных сигналов, объединенных в блоки по времени. Такой поток данных может составлять до 20 ГБ/секунду, что, при планируемых режимах работы ускорительного комплекса и детектора, будет составлять до 200 ПБ/год. С целью сокращения объема данных для долговременного хранения и последующего анализа планируется провести их первичную обработку с использованием специализированной вычислительной системы — SPD Online filter.

1 Организация SPD Online Filter

SPD Online Filter — это высокопроизводительная вычислительная система для высокопропускной обработки данных. Основной целью данной системы является быстрая обработка событий и сокращение объема данных для долговременного хранения.

В качестве входных данных система получает файлы «сырых» данных в формате, определяемом электроникой и системой DAQ (Data Acquisition System). Результатом обработки данных является набор реконструированных событий, содержащий также исходную информацию.

1.1 Требование к SPD Online Filter

Данная вычислительная система должна выполнять следующие задачи:

- раскодирование данных, полученных от DAQ;
- выделение событий;
- фильтрация «скучных» событий в соответствии с заданными физическими критериями;
- упорядочивание выходных данных, объединение событий в файлы, а файлы в наборы данных для дальнейшей обработки;
- подготовка данных для системы контроля качества данных.

1.2 Архитектура SPD Online Filter

Для выполнения данных задач предполагается следующая архитектура системы (рис. 1):

- высокоскоростное хранилище входных данных, полученных с помощью системы сбора данных (DAQ);
- буфер для промежуточных данных и данных, подготовленных к передаче в долгосрочное хранилище и будущей обработке;

- комплекс промежуточного программного обеспечения для автоматизации рабочего процесса, в который будут входить следующие компоненты:
  - *Система управления данными* (регистрация новых данных, каталогизация/структуризация, контроль целостности);
  - *Система управления процессами обработки* (формирование и контроль исполнения этапов обработки данных);
  - *Система управления нагрузкой* (реализация этапов обработки, посредством формирования и выполнения необходимого количества задач для обработки набора данных);
    - *Pilot* (агентское приложение, работающее на вычислительном узле и исполняющее задачи, поставляемые от системы управления нагрузкой).

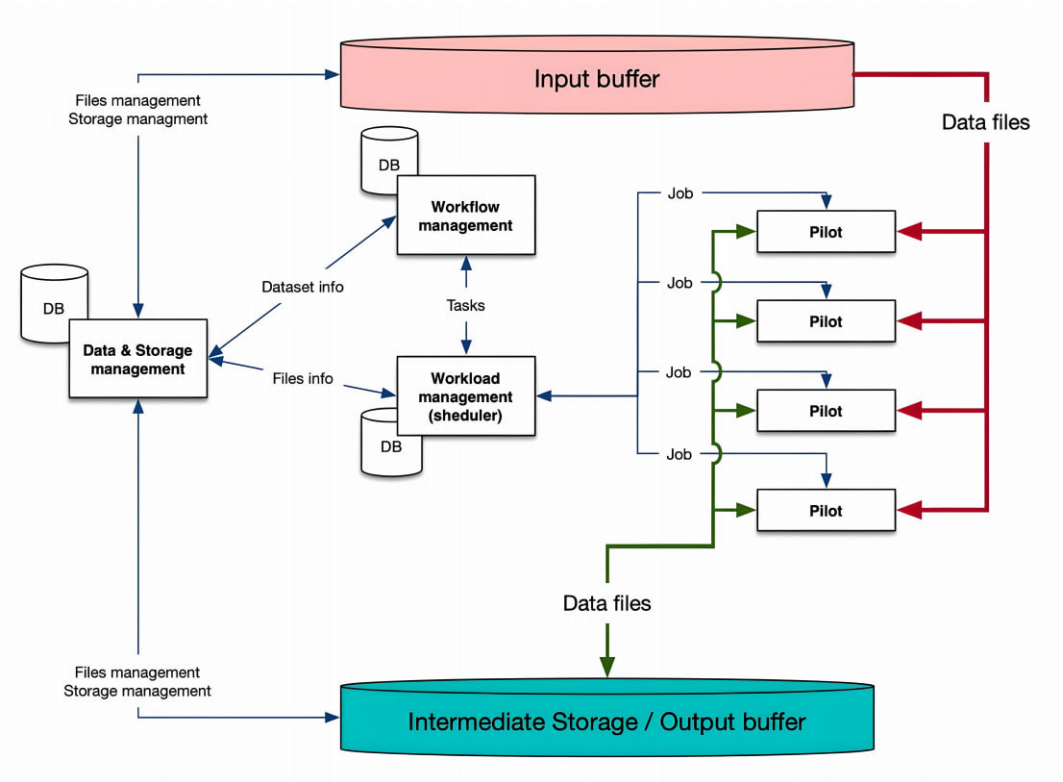


Рисунок 1 – Архитектура SPD Online Filter.

2 Система управления данными

Так как в процессе обработки данных образуется большой объем вторичных данных, то нам необходима специальная система для управления этими данными. Такой системой является система управления данными.

2.1 Требования к системе

Система должна обеспечивать контроль над хранением, организацией, а также целостностью данных.

Для того чтобы система могла узнать о существовании файла в хранилище, необходим интерфейс для регистрации файлов, который включает в себя получение информации о местоположении файла (имя, физический путь, метаданные) и внесение его в каталог со ссылкой на требуемый набор данных.

Из функциональности выше следует то, что информация о файлах и наборах должна храниться в каталоге. Это означает наличие интерфейса к каталогу данных, через который можно размещать информацию о файлах, запрашивать информацию из каталога, удалять информацию в каталоге.

Так как каждый шаг обработки данных оперирует не самим файлом, а набором из них, то система управления данными должна предоставлять возможность управления наборами (датасетами). Сюда входят следующие функции: создать датасет, добавить файл в датасет, закрыть датасет, удалить датасет, дать информацию о содержимом датасет (файлах в датасете).

Для корректного функционирования всей системы требуются фоновые сервисы, которые бы осуществляли удаление файлов в хранилищах, контроль целостности файлов и общий контроль использования хранилища.

2.2 Архитектура

Концептуальная архитектура системы управления данными с учетом требуемой функциональности представлена на рисунке 2.

Система состоит из трех независимых друг от друга сервисов [3]:

- **dsm-register**. Сервис, принимающий в асинхронном режиме (через

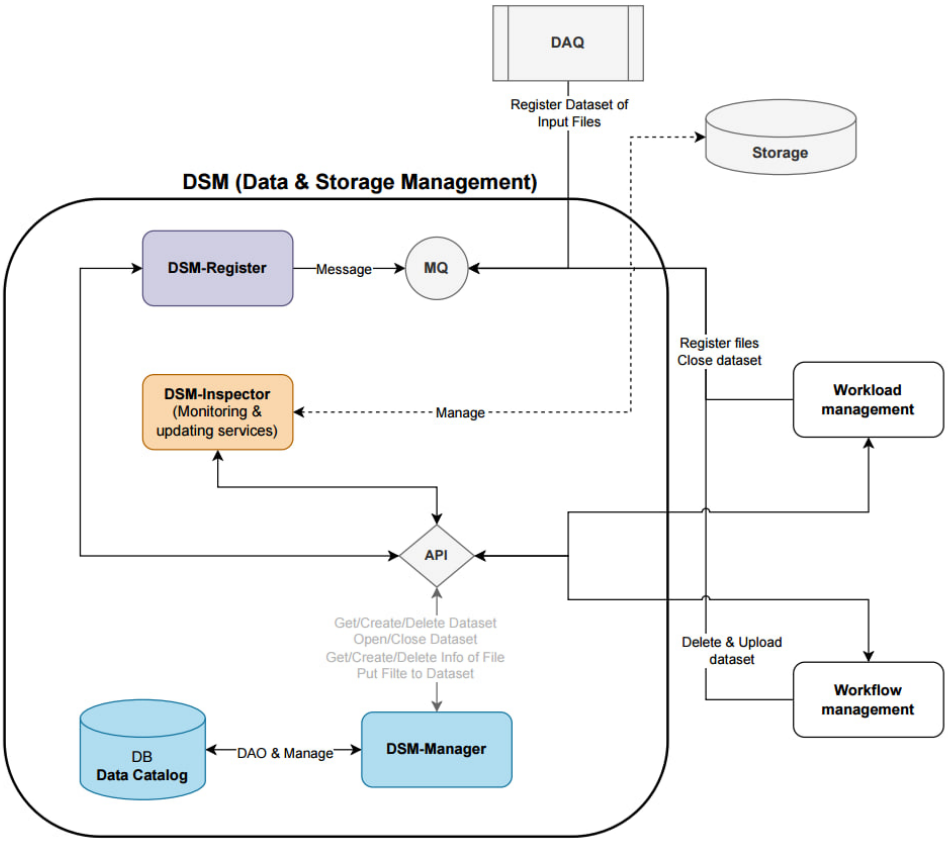


Рисунок 2 – Концептуальная архитектура системы управления данными.

очередь сообщений) заявки на добавление/удаление данных в системе. При обработке заявок сервис вносит изменения в каталог данных через API сервиса dsm-manager;

- **dsm-manager**. Сервис, предоставляющий REST API к каталогу данных (размещение данных в каталоге, обращение к каталогу, изменение данных в каталоге). Он нужен как для внутреннего функционирования системы, так и для внешнего взаимодействия;

- **dsm-inspector**. Набор фоновых сервисов для мониторинга и контроля состояния данных в хранилище (проверка целостности файлов, контроль использования хранилища, удаление файлов на хранилищах).

В тех случаях, где требуется мгновенный ответ от сервиса, используется HTTP протокол обмена данными. Для остальных случаев используется AMQP протокол для асинхронной передачи данных в виде сообщений.

3 Модель данных

Исходя из таких критериев, как качество поддержки, функциональность, широкий круг пользователей, доступность, в качестве СУБД был выбран PostgreSQL 12 [4].

Концептуальная модель данных / ER-диаграмма представлена на рисунке 3. Сюда вошел следующий набор таблиц:

- *DAT_FILE* — каталог файлов, обрабатываемых системой;
- *DAT_DATASET* — каталог наборов файлов, созданных системой;
- *DAT_FILE_HISTORY* и *DAT_DATASET_HISTORY* — архивные таблицы по файлам и наборам;
- *DAT_STORAGE* — информация о хранилищах;
- *DIC_FILE_STATUS* и *DIC_DATASET_STATUS* — справочники, хранящие возможные статусы по файлам и наборам соответственно.

Каталог файлов является ключёвой таблицей в базе данных. В её состав атрибутов вошла та информация, которой достаточно для покрытия следующих целей:

1. Идентификация файла на хранилище (имя файла, путь на хранилище, идентификатор хранилища);
2. Контроль целостности файла (размер файла и его контрольная сумма);
3. Фиксация жизненного цикла файла (статус файла).

Помимо этого предусмотрена возможность принадлежности файла к одному или нескольким наборам. Это осуществляется с помощью ассоциативной таблицы, реализующей отношение «многие ко многим» [5].

Каталог наборов служит для хранения логических группировок файлов, не релевантных к физическому расположению. Каждый набор имеет уникальное наименование, определяемое логикой группировки. Также в

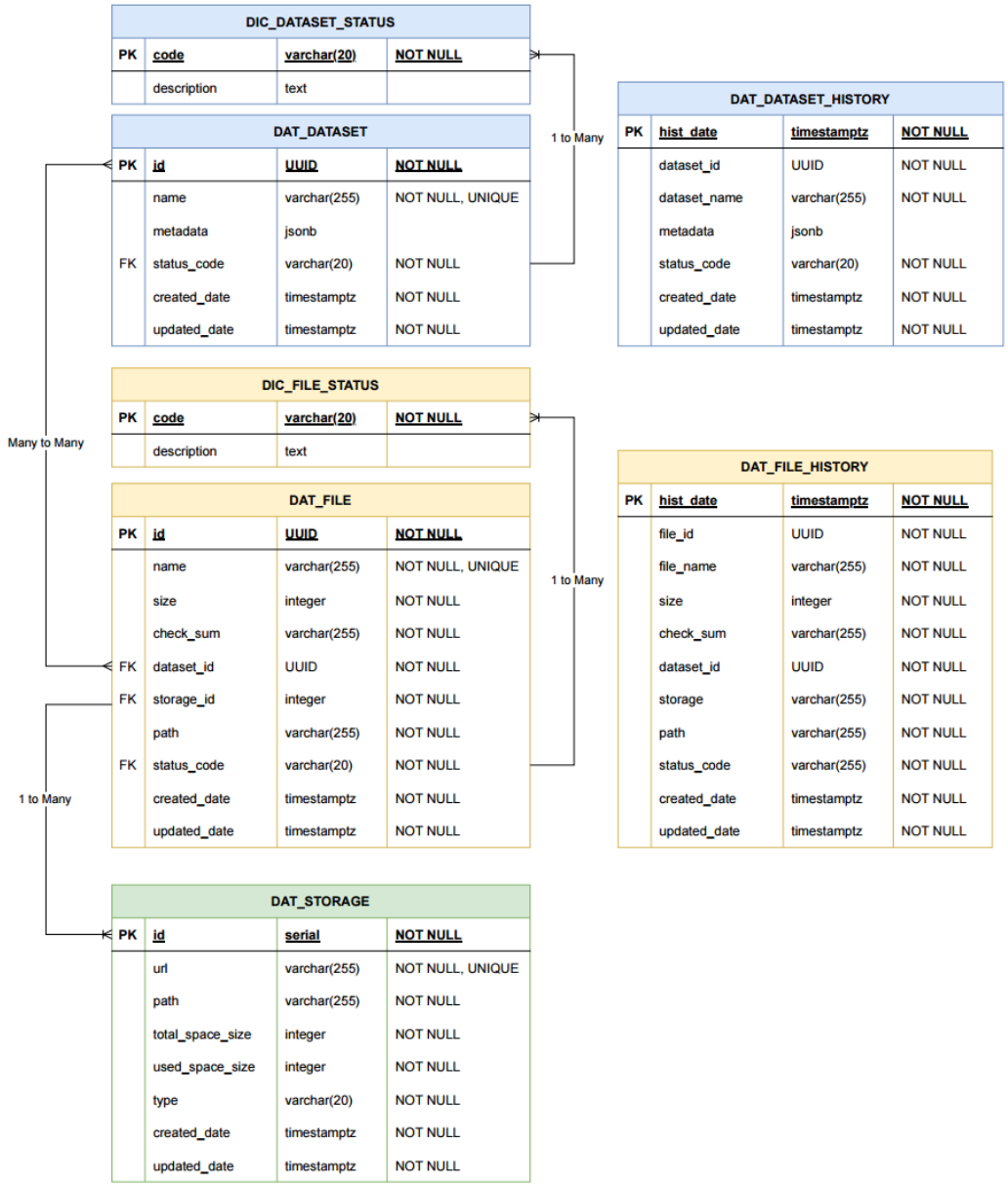


Рисунок 3 – Концептуальная модель данных.

данном каталоге может быть размещена дополнительная неструктурированная мета-информация о наборе в JSON поле (например, в случае входного набора файлов, поступающих с DAQ, это может быть информация о временном срезе данных: номер фрейма, номер рана и т.п.). Набор, как и файл, имеет свой жизненный цикл, поэтому для его фиксации также имеется поле со статусом.

Хранилища помимо адреса могут характеризоваться также дополнительной важной информацией:

1. Тип хранилища (например, входное, промежуточное, выходное);

2. Полный путь до хранилища (внешний и внутренний адрес);

3. Состояние хранилища (размер всего дискового пространства и уже занятого)

Т.к. изменения в каталоге файлов и наборов может происходить достаточно часто, были введены дополнительные **history таблицы** для фиксации предыдущих состояний данных. Это полезно для последующего анализа корректности работы системы. Заполнение такого рода таблиц предполагается осуществлять автоматически при помощи триггера.

Отдельные **справочные таблицы** предназначены для строгого контроля возможных статусов файлов и наборов (см. таблицы 1, 2).

Код статуса	**Описание**
CREATED	Файл добавлен в систему
DAMAGED	Файл поврежден
TO_DELETE	Файл с пометкой «на удалении»
UPLOADING	Файл выгружается
DELETED	Файл удален из системы

Таблица 1: Справочник статусов файла (DIC_FILE_STATUS)

Код статуса	**Описание**
OPEN	Набор открыт
CLOSED	Набор закрыт
FROZEN	Набор временно «заморожен»
TO_UPLOAD	Набор с пометкой «на выгрузку»
UPLOADING	Набор выгружается
TO_DELETE	Набор с пометкой «на удалении»
DELETED	Набор удален из системы

Таблица 2: Справочник статусов набора (DIC_DATASET_STATUS)

4 Требования к сервисам

4.1 Сервис dsm-register

Сервис должен слушать очередь сообщений и обрабатывать заявки на добавление/удаление данных в системе. В таблице 3 представлены шлюзы приёма сообщений. В качестве AMQP-брокера, который осуществляет маршрутизацию и подписку на нужные очереди, выбран RabbitMQ [6].

Exchange	**Routing Key**	**Назначение**
dsm.register (direct)	file.input	Приём информации о поступивших файлах на входной буфер
	file.process	Приём информации о новых файлах, полученных в процессе обработки
	dataset.close	Приём заявки на закрытие набора файлов
	dataset.upload	Приём заявки на выгрузку файлов в наборе во внешнее хранилище
	dataset.delete	Приём заявки на удаление файлов в наборе на внутреннем хранилище

Таблица 3: Перечень шлюзов приёма сообщений сервиса dsm-register

Шлюз **dsm.register.file.input** принимает сообщения вида: путь до хранилища, путь к файлу на хранилище, имя файла, его размер и контрольная сумма. Далее, выполняются следующие шаги:

1. Заводится набор по номеру frame-а со статусом OPEN;
2. Регистрируются файлы с привязкой к этому набору. Устанавливается первичный статус CREATE.

Шлюз **dsm.register.file.process** принимает сообщения вида: идентификатор набора, информация о файлах в наборе (путь к файлу на хранилище, включая имя файла, идентификатор хранилища, размер файла и его контрольная сумма). Далее происходит регистрация новых файлов (в статусе CREATE) с привязкой к указанному набору.

Шлюз **dsm.register.dataset.close** принимает сообщения вида: идентификатор набора и контрольный список файлов (имена файлов). Далее, выполняются следующие шаги:

1. Запрашивается список зарегистрированных файлов в указанном наборе;
2. Если данный список совпадает с контрольным списком, то у набора устанавливается статус CLOSE. Иначе сообщение возвращается обратно в очередь для отложенного исполнения.

Шлюзы **dsm.register.dataset.upload** и **dsm.register.dataset.delete** принимают только идентификатор набора. Далее, выполняются следующие шаги:

1. Запрашивается текущий статус набора;
2. Если набор находится в статусе OPEN, то сообщение возвращается обратно в очередь для отложенного исполнения;
3. Иначе устанавливается статус TO_UPLOAD и TO_DELETE соответственно.

Также после регистрации первичных и вторичных файлов необходимо дополнительно отправлять сообщения в соответствующие очереди для информирования других подсистем. Шлюзы отправки сообщений представлен в таблице 4.

Exchange	**Routing Key**	**Назначение**
dsm.register (direct)	file.process.reply	Отправка информации о статусе регистрации файлов, полученных в процессе обработки
	dataset.input	Отправка информации о входных датасетах

Таблица 4: Перечень шлюзов отправки сообщений сервиса dsm-register

Шлюз **dsm.register.file.process.reply** отправляет сообщения вида: статус регистрации файла (SUCCESS или ERROR), детали регистрации (если SUCCESS – полное имя зарегистрированного файла, если ERROR – полное имя файла и тип возникшей ошибки).

Шлюз **dsm.register.dataset.input** отправляет сообщения с информацией о входных датасетах (имя и id).

4.2 Сервис dsm-manager

В таблице 5 представлен перечень REST API к каталогу данных в системе, который должен предоставлять сервис dsm-manager. Часть из них нужны для внутреннего функционирования системы управления данными, остальная часть для внешнего взаимодействия.

Функциональность	**URL**	**Контракт запроса**	**Контракт ответа**
Внутреннее API			
Управление информацией о наборах в каталоге			
Создать набор	POST /dataset	Информация о наборе	ID набора
Получить набор	GET /dataset/<id>	ID набора	Информация о наборе
Изменить набор	PUT /dataset/<id>	ID набора и изменения	Обновленная информация о наборе
Удалить набор	DELETE /dataset/<id>	ID набора	-
Получить список наборов	GET /dataset	-	Информация о наборах
Управление информацией о файлах в каталоге			
Добавить файл	POST /file	Информация о файле и принадлежность к набору	ID файла
Получить файл	GET /file/<id>	ID файла	Информация о файле
Изменить файл	PUT /file/<id>	ID файла и изменения	Обновленная информация о файле
Удалить файл	DELETE /file/<id>	ID файла	-
Получить список файлов	GET /file	-	Информация о файлах
Управление информацией о хранилище			
Добавить хранилище	POST /storage	Информация о хранилище	ID хранилища
Получить хранилище	GET /storage/<id>	ID хранилища	Информация о хранилище
Изменить хранилище	PUT /storage/<id>	ID хранилища и изменения	Обновленная информация о хранилище
Удалить хранилище	DELETE /storage/<id>	ID хранилища	-
Получить список хранилищ	GET /storage	-	Информация о хранилищах
Получение информации для мониторинга			
Получение списка наборов, к которым принадлежит файл	GET /dataset/?file_id=<id>	ID файла	Информация о наборах
Список наборов в определенном статусе	GET /dataset/?status=<>	Статус набора	Информация о наборах
Список файлов в определенном статусе	GET /file/?status=<>	Статус файла	Информация о файлах
Поиск информации о файле по имени	GET /file/file_name/<name>	Имя файла	Информация о файле
Внешнее API			
Взаимодействие с системой управления нагрузкой			
Содержание набора	GET /file/?dataset_id=<id>	ID набора	Информация о файлах
Взаимодействие с системой управления процессами обработки			
Некоторые API управления наборами (создание выходного набора, получение статуса выходного набора)			

Таблица 5: Описание REST API сервиса dsm-manager

4.3 Сервис dsm-inspector

Сервис, состоящий из набора фоновых задач для мониторинга состояния системы:

- Проверка целостности файлов;
- Удаление датасетов и файлов;
- Контроль использования хранилища;
- Контроль выгрузки данных.

Проверка целостности файлов является сервисом мониторинга и выполняет проверку раз в 5 часов. Алгоритм работы следующий:

1. Получаем список наборов в статусе CLOSED;
2. По каждому набору получаем его содержимое (информацию о файлах);
3. Формируем список уникальных файлов (по ID);
4. По каждому файлу:
    - проверяем наличие файла на хранилище;
    - проверяем размер файла;
    - проверяем контрольную сумму;
    - если какая-то из проверок не пройдена, устанавливаем файлу статус DAMAGED.
5. Если какой-либо файл в наборе поврежден, устанавливаем набору статус FROZEN.

Удаление датасетов и файлов является сервисом исполнения заявок на удаление наборов и выполняется при помощи трех процессов:

- Определения списка файлов, подлежащих удалению (хранимая процедура в БД):

1. Определяем список наборов в статусе TO_DELETE;
2. Для каждого набора определяем список файлов;
3. Для каждого файла проверяем, принадлежит ли он еще какому-нибудь набору:
   - если нет, то устанавливаем ему статус TO_DELETE;
   - если да, то отцепляем файл от данного датасета (удаляем соответствующую запись в таблице).

- Удаление файлов в датасетах:

1. Получаем список файлов со статусом TO_DELETE;
2. Удаляем каждый файл с хранилища и помечаем его в каталоге статусом DELETED (если такого файла на хранилище нет, все равно устанавливаем ему в каталоге статус DELETED).

- Удаление датасетов (хранимая процедура в БД):

1. Получаем список датасетов со статусом TO_DELETE;
2. Для каждого датасета смотрим, все ли файлы находятся в статусе DELETED;
3. Если да, то помечаем датасет статусом DELETED.

Контроль использования хранилища является сервисом мониторинга и имеет два сценария проверки:

- Мониторинг входного и выходного хранилища на предмет «тёмных» файлов:

1. Получаем список файлов на хранилище;
2. Ищем файл в каталоге по имени;
3. Если файл в каталоге не найден, то добавляем его в таблицу для «тёмных» файлов.

- Мониторинг заполненности хранилища:

1. Получаем информацию по хранилищам (url, path);
2. Проверяем размер занятого пространства на хранилищах и обновляем эту информацию в каталоге.

Контроль выгрузки данных является сервисом исполнения заявок на выгрузку файлов в наборе во внешнюю систему и так же выполняется при помощи двух параллельных процессов:

- Постановка задач на выгрузку:
  1. Получаем список наборов со статусом TO_UPLOAD;
  2. По каждому набору получаем его содержимое;
  3. Создаём задачу во внешней системе на выгрузку набора (по ID/имени набора);
  4. Устанавливаем для каждого файла в наборе статус UPLOADING;
  5. Устанавливаем статус набору UPLOADING.
- Мониторинг выгрузки:
  1. Получаем список наборов со статусом UPLOADING;
  2. По ID/имени набора получаем статус выполнения задачи на выгрузку;
  3. Если статус «УСПЕШНО», то помечаем набор статусом TO_DELETE.

5 Результаты

5.1 Задачи

В ходе работы были выполнены следующие задачи:

1. Добавление дополнительного функционала для взаимодействия dsm-register с системой управления процессами обработки:
    - Настроить механизм отправки сообщений с информацией о входных датасетах в новую очередь *dsm.register.dataset.input* (предварительно создать её);
    - Реализовать прием заявок на удаление датасета:
        - Создать новый consumer для очереди *dsm.register.dataset.delete*
        - Написать обработчик сообщений, который будет запрашивать статус набора и, либо отправлять сообщение обратно в очередь (если статус OPEN), либо устанавливать набору статус TO_DELETE.
2. Частичная реализация сервиса dsm-inspector:
    - Разработка сервиса проверки целостности файлов;
        - Дополнительно необходимо было добавить в dsm-manager возможность получения списка наборов, к которым принадлежит файл с данным ID.
    - Разработка сервиса удаления датасетов и файлов:
        - Написать две процедуры, которые будут храниться непосредственно в базе данных и запускаться по расписанию каждый час. Первая процедура должна помечать файлы, подлежащие удалению, статусом TO_DELETE, вторая — удалять датасеты (помечать их статусом DELETED);
        - Дополнительно необходимо было добавить в dsm-manager возможность получения списка файлов по статусу.

5.2 Механизм отправки сообщений о входных наборах

Для начала проверим работоспособность механизма отправки информации о входных датасетах. Необходимо было создать новую очередь, в которую отправлялись бы данные сообщения — *dsm.register.dataset.input*. На рисунке 4 представленные сконфигурированные очереди RabbitMQ.

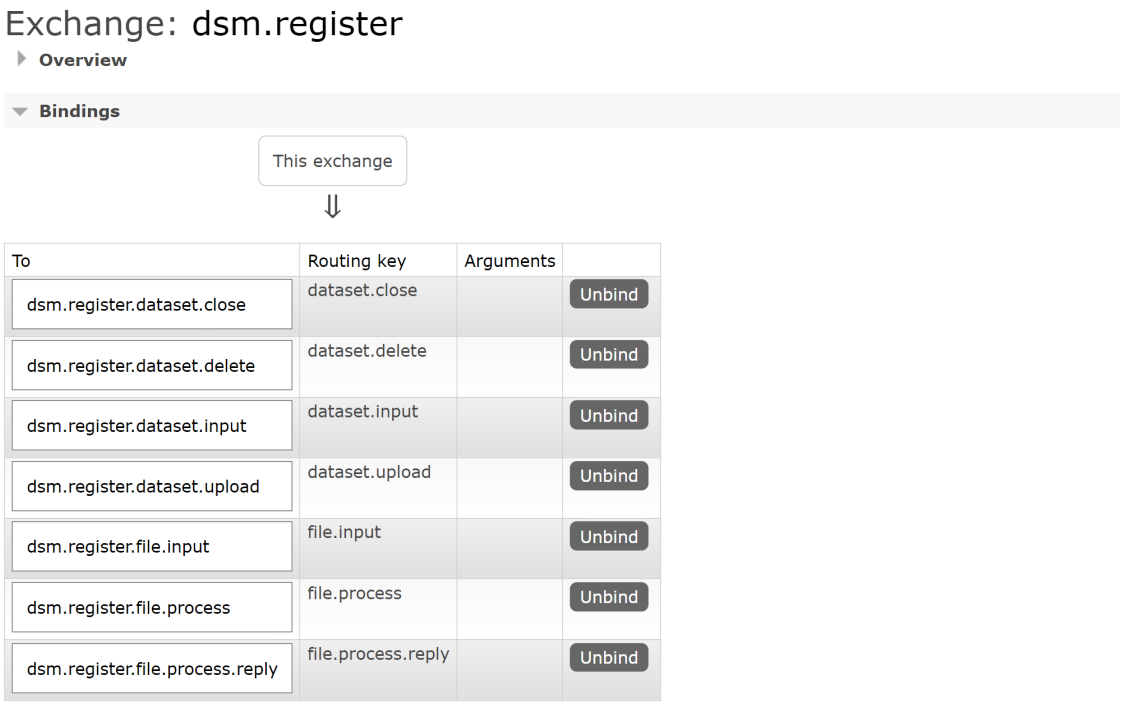


Рисунок 4 – Сконфигурированные очереди RabbitMQ.

Далее был написан примитивный клиент, который создает файлы на входном хранилище и направляет в очередь *dsm.register.file.input* сообщение с информацией о них (рисунок 5a).

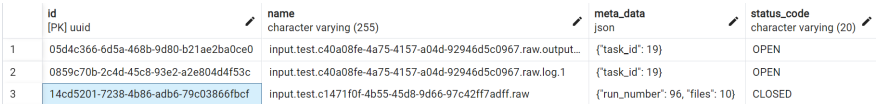
Сервис dsm-register получает данное сообщение, создает новый набор и регистрирует файлы в каталоге с привязкой к данному набору. После завершения регистрации всех файлов набор закрывается (рисунок 5b). И информация об этом наборе направляется в очередь *dsm.register.dataset.input* (рисунок 5c) для информирования системы управления процессами обработки.

(a) Сообщение о входных файлах.

(b) Лог о завершении регистрации входных файлов и закрытии созданного датасета.

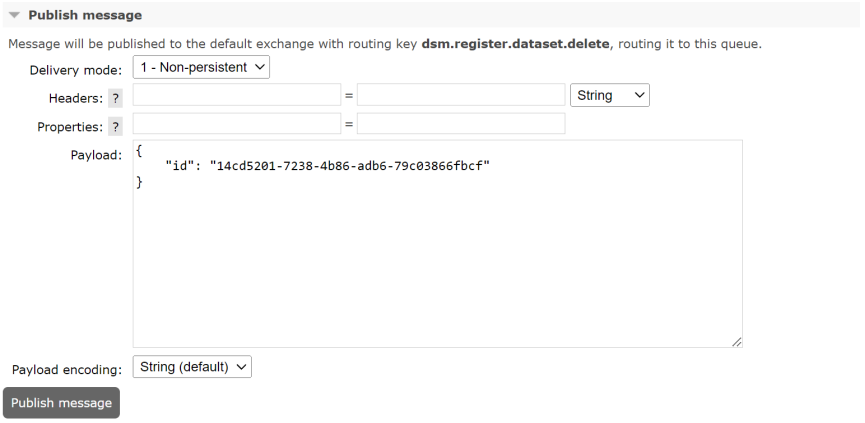
(c) Сообщение с информацией о созданном датасете.

Рисунок 5 – Проверка работоспособности механизма регистрации первичных файлов и отправки сообщений о входных наборах.



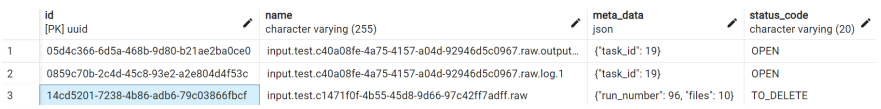
	id [PK] uuid	name character varying (255)	meta_data json	status_code character varying (20)
1	05d4c366-6d5a-468b-9d80-b21ae2ba0ce0	input.test.c40a08fe-4a75-4157-a04d-92946d5c0967.raw.output...	{"task_id": 19}	OPEN
2	0859c70b-2c4d-45c8-93e2-a2e804d4f53c	input.test.c40a08fe-4a75-4157-a04d-92946d5c0967.raw.log.1	{"task_id": 19}	OPEN
3	14cd5201-7238-4b86-adb6-79c03866fbcf	input.test.c1471f0f-4b55-45d8-9d66-97c42ff7adff.raw	{"run_number": 96, "files": 10}	CLOSED

(a) Информация о датасете в статусе CLOSED.



(b) Заявка на удаление датасета в статусе CLOSED.

(c) Лог с информацией о получении сообщения и изменения статуса набора на TO_DELETE.



	id [PK] uuid	name character varying (255)	meta_data json	status_code character varying (20)
1	05d4c366-6d5a-468b-9d80-b21ae2ba0ce0	input.test.c40a08fe-4a75-4157-a04d-92946d5c0967.raw.output...	{"task_id": 19}	OPEN
2	0859c70b-2c4d-45c8-93e2-a2e804d4f53c	input.test.c40a08fe-4a75-4157-a04d-92946d5c0967.raw.log.1	{"task_id": 19}	OPEN
3	14cd5201-7238-4b86-adb6-79c03866fbcf	input.test.c1471f0f-4b55-45d8-9d66-97c42ff7adff.raw	{"run_number": 96, "files": 10}	TO_DELETE

(d) Проверка изменения статуса набора в базе данных.

Рисунок 6 – Проверка механизма приема и обработки заявок на удаление датасета в статусе CLOSED.

5.3 Прием и обработка заявок на удаление датасета

Далее необходимо было реализовать прием заявок на удаление набора. Все подобные заявки проходят через очередь *dsm.register.dataset.delete*.

Сервис dsm-register при получении заявки должен проверять статус датасета, который мы хотим удалить. Если статсу OPEN, то заявка должна быть возвращена в очередь для отложенной обработки.

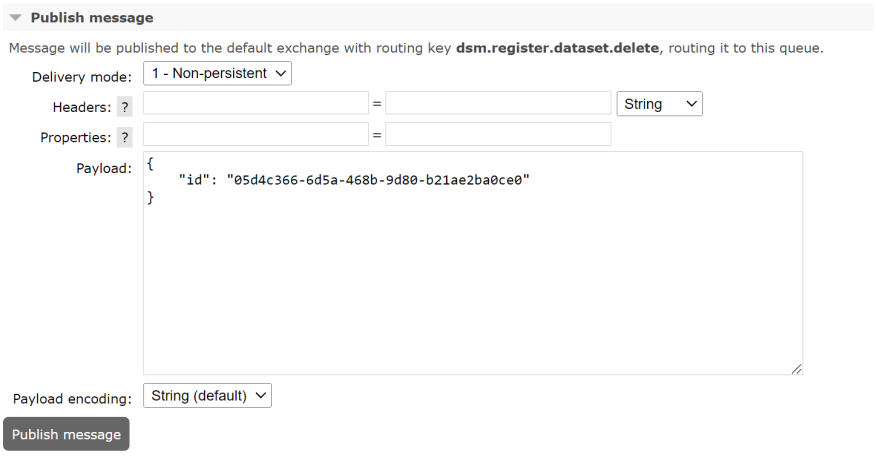
Выберем какой-нибудь набор в статусе CLOSED (рисунок 6a) и через панель администрирования RabbitMQ отправим заявку на удаление выбранного датасета (рисунок 6b).

Сервис dsm-register получает сообщение из очереди, обрабатывает его и изменяет статус набора на TO_DELETE (рисунок 6c-6d).

Если мы теперь захотим отправить заявку на удаление открытого набора (рисунок 7a-7b), то увидим, что после проверки статуса датасета, сервис возвращает сообщение обратно в очередь (рисунок 7c).

	id [PK] uuid	name character varying (255)	meta_data json	status_code character varying (20)
1	05d4c366-6d5a-468b-9d80-b21ae2ba0ce0	input.test.c40a08fe-4a75-4157-a04d-92946d5c0967.raw.output...	{"task_id": 19}	OPEN

(a) Информация о датасете в статусе OPEN.



(b) Заявка на удаление датасета в статусе OPEN.

```
app-1  | 2025-01-18 16:34:40 INFO: [DSM-REGISTER-I004] [CONSUMER-HANDLER] Received message DatasetDeleteDto(id=UUID('05d4c366-6d5a-468b-
9d80-b21ae2ba0ce0')) from queue '.dsm.register.dataset.delete' with delivery tag 2547 [in /src/app/executer/rabbit/consumers/base/consum
er_handler.py:50
app-1  | 2025-01-18 16:34:40 INFO: Processing msg. [in /src/app/executer/services/processor/dataset_delete_processor.py:23
app-1  | 2025-01-18 16:34:40 INFO: Dataset ID=05d4c366-6d5a-468b-9d80-b21ae2ba0ce0 OPEN. The message returned to the queue for deferred
processing. [in /src/app/executer/services/processor/dataset_delete_processor.py:28
```

(c) Лог с информацией о получении сообщения и возвращения его в очередь для отложенной обработки.

Рисунок 7 – Проверка механизма приема и обработки заявок на удаление датасета в статусе OPEN.

5.4 Сервис проверки целостности файлов

Теперь проверим, как работает в dsm-inspector проверка целостности файлов. Данный сервис раз в 5 часов должен проходиться по всем закрытым наборам и проверять по каждому файлу его наличие на хранилище, размер и контрольную сумму.

Выберем какой-нибудь файл (рисунок 8a-8b), принадлежащий набору в статусе CLOSED, и в базе данных изменим его размер (рисунок 8c).

	id [PK] uuid	name character varying (255)	path character varying (255)	storage_id uuid	size integer
1	0fb1e1af-5c02-4d17-87a9-64debf5e6a17	input.test.a976a020-3de5-44e2-91ee-319e426eda2f.raw	input_40	b3307ad4-f2b3-4f3a-a390-4f4e2762c620	50

(a) Корректная информация о файле в БД.

(b) Полная информация о выбранном файле в БД.

	id [PK] uuid	name character varying (255)	path character varying (255)	storage_id uuid	size integer
1	0fb1e1af-5c02-4d17-87a9-64debf5e6a17	input.test.a976a020-3de5-44e2-91ee-319e426eda2f.raw	input_40	b3307ad4-f2b3-4f3a-a390-4f4e2762c620	100

(c) Измененная информация (размер) о файле в БД.

Рисунок 8 – Изменение информации о размере файла в базе данных.

Проверим также, что выбранный файл действительно принадлежит закрытому набору (рисунок 9).

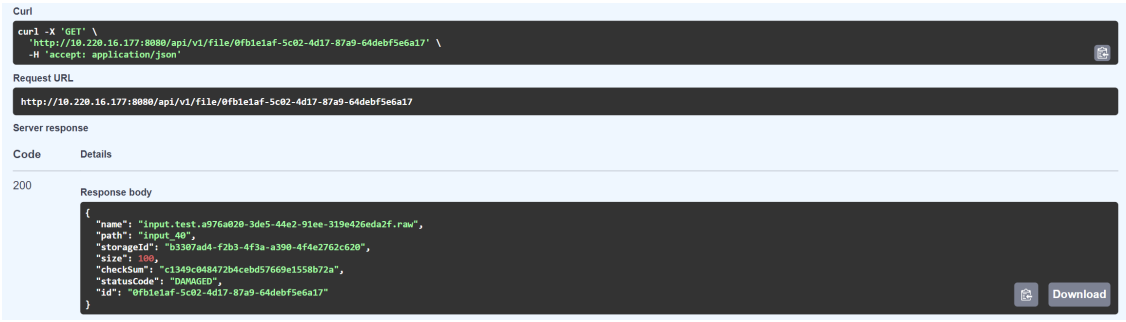
Рисунок 9 – Информация о наборе, которому принадлежит выбранный файл.

После проверки целостности всех файлов видим, что выбранный нами файл оказывается поврежден и что соответствующий набор переведен в статус FROZEN (рисунок 10).

```
integrity-inspector-1  | 2025-01-19 13:31:47 INFO: File /data/SPDOF-buffers/input/input_40/input.test.a976a020-3de5-44e2-91ee-319e426eda
2f.raw DAMAGED! [in /src/files_integrity_inspector/file_integrity_inspector.py:77
integrity-inspector-1  | 2025-01-19 13:31:47 INFO: HTTP Request: PUT http://app:8080/api/v1/dataset/f61828be-64b5-44e8-9d18-a1a22068094d
 "HTTP/1.1 200 OK" [in /src/.venv/lib/python3.11/site-packages/httpx/_client.py:1038
integrity-inspector-1  | 2025-01-19 13:31:47 INFO: Dataset ID=f61828be-64b5-44e8-9d18-a1a22068094d FROZEN [in /src/files_integrity_inspe
ctor/file_integrity_inspector.py:89
```

Рисунок 10 – Лог с информацией о повреждении файла и изменении статуса набора на FROZEN.

Далее через API, которое предоставляет dsm-manager к базе данных, проверяем, действительно ли произошло изменение статуса у файла и набора. Как видно из рисунка 11, механизм действительно отработал.



(a) Информация о выбранном файле.

(b) Информация о соответствующем наборе.

Рисунок 11 – Проверка изменения статусов файла и датасета после отработки сервиса проверки целостности файлов.

Здесь рассматривался только случай несовпадения фактического размера файла с его каталожной записью, но это справедливо и в случаях контрольной суммый файла и его наличия на хранилище.

5.5 Сервис удаления датасетов и файлов

В заключение проверим часть сервиса dsm-inspector, которая отвечает за удаление датасетов и файлов.

Запрос История запросов

```
SELECT dat_file.id, dat_file.name, dat_file.status_code, dat_file.path FROM dat_dataset
JOIN dat_association_file_dataset fd on fd.dataset_id = dat_dataset.id
JOIN dat_file on fd.file_id = dat_file.id
WHERE dat_dataset.status_code = 'TO_DELETE'
```

Data Output Сообщения Notifications

	id [PK] uuid	name character varying (255)	status_code character varying (20)	path character varying (255)
1	c81794fe-c57c-485c-89c7-0ace02f81883	input.test.41dad14a-e58c-4cd0-979a-16ee3e1cdb03.r...	CREATED	input_4
2	aa20e90d-6892-4d1a-aad1-1f3c5a7e4dc2	input.test.08850bb9-ae56-460e-875e-486f166c7533.r...	CREATED	input_4
3	b41ef691-3a5d-4d17-90d9-715a01e109a7	input.test.297eb0ea-c6e2-4261-acdf-b0d6f1c87482.ra...	CREATED	input_4
4	baa79005-1115-4bef-a82c-3b342c2213f0	input.test.8e607ad1-c88a-4e95-ba61-b1d539282662.r...	CREATED	input_4
5	6fa4d091-f983-4e21-92b8-ce68ea020732	input.test.d8a5c878-1ebe-4828-84b7-b37056bf6ac5.r...	CREATED	input_4
6	4255668b-26f2-4be7-98d0-64d82832c3ac	input.test.faa4e27f-ae2a-49d8-8934-5bfaa3102709.raw	CREATED	input_4
7	ec5ba1bf-b30b-47dc-8bc7-c76689bb58e1	input.test.8c04f7d8-2f26-4cc3-a81e-1b0dc796239e.ra...	CREATED	input_4
8	1e87276a-0462-4964-803e-7b6f96ea018d	input.test.3ea2562e-aa73-4c5e-8911-576db19d7fc5.r...	CREATED	input_4
9	8e238086-b493-459e-a567-627e00ab08e4	input.test.0aec3382-efe0-4986-8237-f07bc545248c.ra...	CREATED	input_4
10	760ecdb0-6f41-4515-b880-c7c21890ed81	input.test.451393ef-5ecb-4ed8-afd4-454acecc92d6.raw	CREATED	input_4

(a) Информация о файлах, принадлежащих набору в статусе TO_DELETE.

Запрос История запросов

```
SELECT dat_file.id, dat_file.name, dat_file.status_code, dat_file.path FROM dat_dataset
JOIN dat_association_file_dataset fd on fd.dataset_id = dat_dataset.id
JOIN dat_file on fd.file_id = dat_file.id
WHERE dat_dataset.status_code = 'TO_DELETE'
```

Data Output Сообщения Notifications

	id [PK] uuid	name character varying (255)	status_code character varying (20)	path character varying (255)
1	c81794fe-c57c-485c-89c7-0ace02f81883	input.test.41dad14a-e58c-4cd0-979a-16ee3e1cdb03.r...	TO_DELETE	input_4
2	aa20e90d-6892-4d1a-aad1-1f3c5a7e4dc2	input.test.08850bb9-ae56-460e-875e-486f166c7533.r...	TO_DELETE	input_4
3	b41ef691-3a5d-4d17-90d9-715a01e109a7	input.test.297eb0ea-c6e2-4261-acdf-b0d6f1c87482.ra...	TO_DELETE	input_4
4	baa79005-1115-4bef-a82c-3b342c2213f0	input.test.8e607ad1-c88a-4e95-ba61-b1d539282662.r...	TO_DELETE	input_4
5	6fa4d091-f983-4e21-92b8-ce68ea020732	input.test.d8a5c878-1ebe-4828-84b7-b37056bf6ac5.r...	TO_DELETE	input_4
6	4255668b-26f2-4be7-98d0-64d82832c3ac	input.test.faa4e27f-ae2a-49d8-8934-5bfaa3102709.raw	TO_DELETE	input_4
7	ec5ba1bf-b30b-47dc-8bc7-c76689bb58e1	input.test.8c04f7d8-2f26-4cc3-a81e-1b0dc796239e.ra...	TO_DELETE	input_4
8	1e87276a-0462-4964-803e-7b6f96ea018d	input.test.3ea2562e-aa73-4c5e-8911-576db19d7fc5.r...	TO_DELETE	input_4
9	8e238086-b493-459e-a567-627e00ab08e4	input.test.0aec3382-efe0-4986-8237-f07bc545248c.ra...	TO_DELETE	input_4
10	760ecdb0-6f41-4515-b880-c7c21890ed81	input.test.451393ef-5ecb-4ed8-afd4-454acecc92d6.raw	TO_DELETE	input_4

(b) Измененная информация о файлах, подлежащих удалению.

Рисунок 12 – Проверка изменения статусов файлов после отработки процедуры, помечающей файлы, подлежащие удалению.

Для начала проверим корректность работы процедуры, которая для каждого набора, подлежащего удалению, проверяет все его файлы на принадлежность другим датасетам. Если такая принадлежность имеется, то файл отцепляется от проверяемого набора. Если нет, то файл переводится

в статус TO_DELETE.

Посмотрим на список файлов, которые привязаны к набору в статусе TO_DELETE (рисунок 12a). Далее запускаем соответствующую процедуру и видим, что у всех файлов из списка изменился статус (рисунок 12b).

Далее сервис получает список всех файлов, подлежащих удалению, а затем в многопоточном режиме удаляет каждый файл с хранилища и изменяет его статус на DELETED. Это мы видим на рисунке 13.

```
deleting-inspector-1  | 2025-01-19 14:17:13 INFO: Foarm file list with status TO_DELETE [in /src/files_deleting_inspector/file_del
ete_inspector.py:32
deleting-inspector-1  | 2025-01-19 14:17:13 INFO: HTTP Request: GET http://app:8080/api/v1/file/?file_status=TO_DELETE "HTTP/1.1 2
00 OK" [in /src/.venv/lib/python3.11/site-packages/httpx/_client.py:1038
deleting-inspector-1  | 2025-01-19 14:17:13 INFO: HTTP Request: GET http://app:8080/api/v1/storage/b3307ad4-f2b3-4f3a-a390-4f4e276
2c620 "HTTP/1.1 200 OK" [in /src/.venv/lib/python3.11/site-packages/httpx/_client.py:1038
deleting-inspector-1  | 2025-01-19 14:17:13 INFO: File /data/SPDOF-buffers/input/input_4/input.test.8c04f7d8-2f26-4cc3-a81e-1b0dc7
96239e.raw DELETED! [in /src/files_deleting_inspector/file_delete_inspector.py:44
```

(a) Лог с информацией об удалении файлов, находящихся в статусе TO_DELETE.

Запрос | История запросов

```
1 SELECT dat_file.id, dat_file.name, dat_file.status_code, dat_file.path FROM dat_dataset
2 JOIN dat_association_file_dataset fd on fd.dataset_id = dat_dataset.id
3 JOIN dat_file on fd.file_id = dat_file.id
4 WHERE dat_dataset.status_code = 'TO_DELETE'
```

Data Output | Сообщения | Notifications

	id [PK] uuid	name character varying (255)	status_code character varying (20)	path character varying (255)
1	c81794fe-c57c-485c-89c7-0ace02f81883	input.test.41dad14a-e58c-4cd0-979a-16ee3e1cdb03.r...	DELETED	input_4
2	aa20e90d-6892-4d1a-aad1-1f3c5a7e4dc2	input.test.08850bb9-ae56-460e-875e-486f166c7533.r...	DELETED	input_4
3	b41ef691-3a5d-4d17-90d9-715a01e109a7	input.test.297eb0ea-c6e2-4261-acdf-b0d6f1c87482.ra...	DELETED	input_4
4	baa79005-1115-4bef-a82c-3b342c2213f0	input.test.8e607ad1-c88a-4e95-ba61-b1d539282662.r...	DELETED	input_4
5	6fa4d091-f983-4e21-92b8-ce68ea020732	input.test.d8a5c878-1ebe-4828-84b7-b37056bf6ac5.r...	DELETED	input_4
6	4255668b-26f2-4be7-98d0-64d82832c3ac	input.test.faa4e27f-ae2a-49d8-8934-5bfaa3102709.raw	DELETED	input_4
7	ec5ba1bf-b30b-47dc-8bc7-c76689bb58e1	input.test.8c04f7d8-2f26-4cc3-a81e-1b0dc796239e.ra...	DELETED	input_4
8	1e87276a-0462-4964-803e-7b6f96ea018d	input.test.3ea2562e-aa73-4c5e-8911-576db19d7fc5.r...	DELETED	input_4
9	8e238086-b493-459e-a567-627e00ab08e4	input.test.0aec3382-efe0-4986-8237-f07bc545248c.ra...	DELETED	input_4
10	760ecdb0-6f41-4515-b880-c7c21890ed81	input.test.451393ef-5ecb-4ed8-afd4-454acecc92d6.raw	DELETED	input_4

(b) Измененная информация об удаленных файлах.

Рисунок 13 – Проверка изменения статусов файлов после отработки сервиса по удалению файлов.

И в конце переходим к процедуре, которая для каждого набора в статусе TO_DELETE проверяет, все ли файлы в нем помечены как удаленные, и, если это так, то удаляет данный датасет (т.е. устанавливает ему статус DELETED). На рисунке 14 представлена информация о наборе до и после срабатывания процедуры.

	id [PK] uuid	name character varying (255)	meta_data json	status_code character varying (20)
1	bf04dfe1-db61-4195-b584-45de6ea8f04d	input.test.d95edab2-8f40-4b67-9fd7-01565512228f.raw	{"run_number": 3, "files": 10}	TO_DELETE

(a) Информация о наборе, к которому привязаны удаленные файлы.

	id [PK] uuid	name character varying (255)	meta_data json	status_code character varying (20)
1	bf04dfe1-db61-4195-b584-45de6ea8f04d	input.test.d95edab2-8f40-4b67-9fd7-01565512228f.raw	{"run_number": 3, "files": 10}	DELETED

(b) Обновленная информация о наборе.

Рисунок 14 – Проверка изменения статуса набора после отработки процедуры, удаляющей датасеты.

Заключение

При проведении экспериментов на коллайдере чрезвычайно важно решить проблему обработки и хранения огромного объема данных, получаемого в результате столкновений.

В связи с этим в контексте данной работы рассмотрен вычислительный комплекс SPD Online Filter, который предназначен для быстрой реконструкции событий и сокращения объема выходных данных без потери их физической ценности. Это является важным шагом в обеспечении эффективной обработки и анализа экспериментальных данных. И, как следствие, является ключом к пониманию спиновой структуры нуклона.

Также была рассмотрена одна из составляющих системы SPD Online Filter — система управления данными. Она состоит из трех микросервисов: dsm-register, dsm-manager и dsm-inspector. Dsm-register принимает заявки на добавление/удаление данных в системе, dsm-manager предоставляет доступ к каталогу данных, а dsm-inspector отвечает за мониторинг состояния системы.

В работе был добавлен дополнительный функционал для взаимодействия сервиса dsm-register с системой управления процессами обработки и был частично реализован сервис dsm-inspector. Также по ходу выполнения данных задач были добавлены дополнительные возможности в сервис dsm-manager.

В дальнейшем планируется:

1. Закончить реализацию сервиса dsm-inspector (добавить контроль выгрузки файлов и контроль использования хранилища);

2. Для сервиса dsm-register реализовать обработку сообщений из оставшихся очередей: dsm.register.dataset.close и dsm.register.dataset.upload.

Список используемых источников

1. *Ashman J.* [и др.]. A Measurement of the Spin Asymmetry and Determination of the Structure Function g(1) in Deep Inelastic Muon-Proton Scattering // Phys. Lett. B / под ред. V. W. Hughes, C. Cavata. — 1988. — Т. 206. — С. 364.
2. *Abazov V. M.* [и др.]. Conceptual design of the Spin Physics Detector. — 2021. — Янв. — arXiv: 2102.00442 [hep-ex].
3. *Tereshenko D.* Designing a Data Management Service in a Specialized Distributed Computing System SPD Online Filter. — 2023.
4. PostgreSQL 12.15 Documentation. — URL: https://www.postgresql.org/docs/12/index.html.
5. Связи между таблицами базы данных. — URL: https://habr.com/ru/articles/488054/.
6. RabbitMQ Documentation. — URL: https://www.rabbitmq.com/docs.