Санкт-Петербургский государственный университет
Факультет Прикладной математики – процессов управления
Кафедра технологии программирования

Сдобникова Анна Михайловна

Магистерская диссертация

Проектирование структуры базы данных для проекта NICA

Направление *02.04.02*
Фундаментальная информатика и информационные технологии
Технологии баз данных

Научный руководитель:
доктор физ.−мат.наук,
профессор Андрианов С.Н.

Рецензент:
кандидат техн.−наук,
старший научный сотрудник
Юдин Иван Павлович

Санкт-Петербург
2020

Оглавление

Введение

Общая структура проекта NICA и особенности его реализации. Актуализация задачи.

В настоящей работе речь пойдет о разработке структуры базы данных для большого научно-исследовательского проекта. Это небольшая, но очень важная часть огромного эксперимента. Для того, чтобы актуализировать задачу и создать переход между самим глобальным проектом и ролью решаемой задачи в нем, ниже приводится достаточно объемное введение.

В России с 2013 года ведется строительство большого ускорительного комплекса NICA (***N**uclotron-based **I**on **C**ollider f**A**cility*) — российского коллайдера протонов и тяжелых ионов (Рис.1). [1]. Комплекс возводится на базе Лаборатории физики высоких энергий (ЛФВЭ) им.В.И.Векслера и А.М.Балдина Объединенного института ядерных исследований (ОИЯИ) в городе Дубна Московской области. Строительство нового ускорителя планируют завершить в 2022 году.

Ускорительный комплекс создаётся с целью исследования области физики частиц в ранее недоступной области параметров и условий эксперимента — получение интенсивных пучков тяжёлых ионов и поляризованных ядер с целью поиска смешанной фазы ядерной материи и исследования поляризационных эффектов [2] в области энергий до 11 ГэВ/н.

Для физики данная область считается одной из наиболее перспективных. Максимальная энергия ускорителей в принципе ограничивается размерами и стоимостью магнитной системы.

Современные ускорители являются огромными дорогостоящими комплексами, требующими больших интеллектуальных, физических и материальных вложений.

Комплекс NICA обеспечит широкий спектр пучков: от протонных и дейтронных, до пучков, состоящих из таких тяжёлых ионов, как ядра золота. Тяжёлые ядра будут ускоряться до энергии вплоть до 4,5 ГэВ/нуклон, протоны — до 12,6 ГэВ.

Планируется осуществлять прикладные и фундаментальные исследования в таких областях науки и технологии, как:

- радиобиология и космическая медицина;
- терапия раковых заболеваний;
- развитие реакторов, управляемых пучком ускорителя («производство энергии» с подкритичной сборкой), и технологий трансмутации отходов ядерной энергетики;
- тестирование радиационной стойкости электронных устройств.

Наиболее важными фундаментальными направлениями исследований, которые будут проводиться с использованием коллайдера NICA являются:

- Природа и свойства сильных взаимодействий между элементарными составляющими Стандартной модели физики частиц — кварками и глюонами;
- Поиск признаков фазового перехода между адронной материей и КГП, поиск новых состояний барионной материи;
- Изучение основных свойств сильного взаимодействия и КГП-симметрии.

Основная схема процесса ускорения частиц предусматривает три стадии:

1) Формирование пучка и его инжекция;
2) Ускорение пучка;
3) Вывод пучка на мишень или соударения встречных пучков в самом ускорителе.

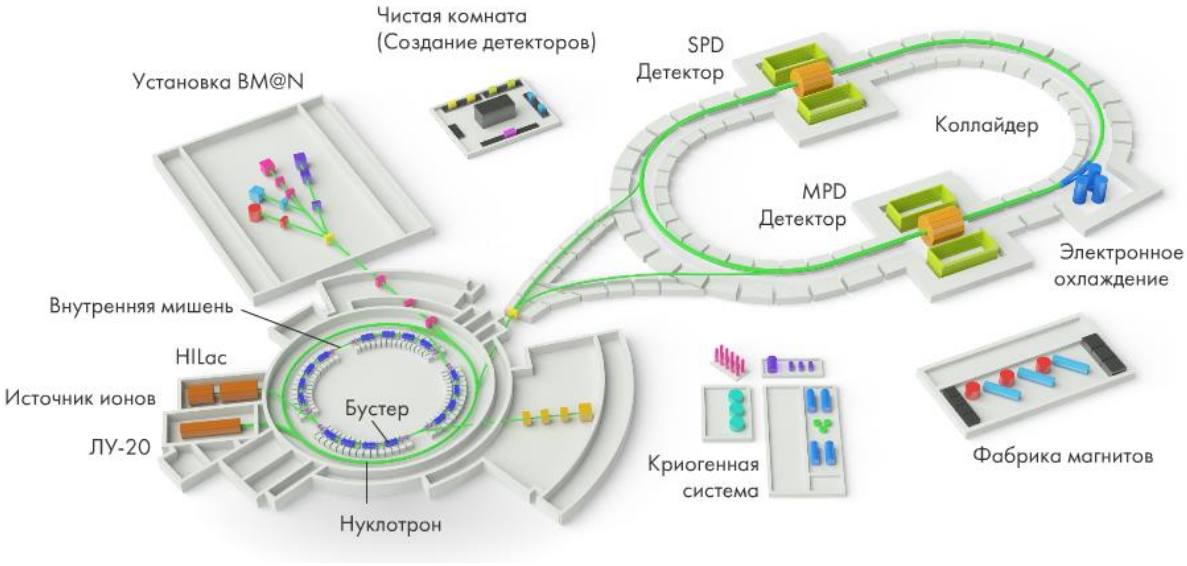

Рис.1. Схема ускорительного комплекса NICA

Основными элементами комплекса NICA являются:

- Инжекционный комплекс поляризованных протонов и дейтронов (источник, линейный ускоритель ЛУ-20)
- Инжекционный комплекс тяжёлых ионов (источник типа КРИОН, линейный ускоритель HILAc)
- Бустерный синхротрон (предускорительное кольцо)
- Нуклотрон (предускорительное кольцо)
- Кольца коллайдера
- Электронное охлаждение
- Криогенный комплекс
- Фабрика магнитов (производство магнитов для комплекса NICA и FAIR
- Чистая комната (производство трековых систем для детекторов)

Детекторы:

- Детектор MPD (Multi-Purpose Detector) предназначен для проведения экспериментов в области релятивистской ядерной физики при столкновениях пучков ядер тяжелых элементов

(золота), ядер тяжелых элементов с протонами и протон-протонных столкновениях.

- Детектор SPD (Spin Physics Detector) предназначен для проведения экспериментов по физике спина при столкновениях пучков ядер легких элементов[3].
- Детектор BM@N (Baryonic Matter at Nuclotron) Целью эксперимента является изучение взаимодействия релятивистских пучков тяжелых ионов с фиксированными мишенями. Является первым экспериментом на ускорительном комплексе NICA–Нуклотрон.

Для успешного проведения эксперимента тщательно должны быть проработаны все этапы подготовки. В первую очередь это касается всех элементов ускорителя.

Принцип действия любого ускорителя основывается на взаимодействии заряженных частиц с электрическим и магнитным полями. Электрическое поле, совершая работу, придает скорость заряженным частицам, в то время как магнитное поле влияет на направление их движения (задает траекторию пучка).

Магнитная система — одна из самых важных в ускорителе[3]. Чем выше энергия частиц, тем труднее пустить их по круговой траектории, и тем сильнее, соответственно, должны быть магнитные поля. Кроме того, частицы нужно фокусировать, чтобы они не отталкивались друг от друга, пока летят. Поэтому магнитные поля должны не только направлять пучок по круговой орбите, но и фокусировать его. Задача магнитов создать «правильное» поле для достижения нужной траектории. За круговое движение отвечают дипольные магниты дугообразной формы, а за фокусирование – объединенные в дуплеты квадрупольные магнитные линзы. Внутри всех магнитов установлены проводники электрического тока больших значений.

Структурными элементами бустера и коллайдера NICA, являются сверхпроводящие (СП) магниты[4]. Сверхпроводящее состояние достигается только при сверхнизких температурах (чем ниже температура магнита, тем меньше сопротивление материала, из которого состоит магнит; таким образом добиваются практически полного отсутствия сопротивления). Для этого в камере ускорителя создается высокий вакуум, и вся структура оснащается системой охлаждения жидким гелием (4.5 К или −269 С°).

Для бустера и коллайдера NICA требуется более 390 сверхпроводящих магнитов. Магнитная система бустера проекта NICA включает 48 квадрупольных сверхпроводящих магнитов. В работе будут описаны методики измерения магнитов[4, 6, 7, 8, 9].

Для успешной работы ускорителя необходимо провести массу сложнейших расчетов для всех его элементов и поставить четкий набор задач, с помощью решения которых возможно достигнуть наиболее точного соответствия в реализации[5]. Т.к. ускорительное оборудование работает с микроструктурой материального мира, то не стоит и рассуждать, насколько важно точнейшее соответствие расчетам на всех этапах подготовки и создания платформы эксперимента. Ошибка может стоить слишком дорого.

Проверка и тестирование всех деталей ускорительного комплекса проводится как на этапах изготовления, так и в процессе юстировки каждого элемента структуры.

В настоящее время для ускорительного комплекса NICA ведется серийное производство сверхпроводящих элементов магнитной оптики ускорительного комплекса (на территории ОИЯИ, г. Дубна). Производство магнитов включает также и проведение набора тестов, вакуумных, электрических, магнитных, а также тренировки сверхпроводящих магнитов.

Также с целью компенсации разброса параметров магнитов, при их расстановке в кольце используются результаты моделирования движения пучков в ускорителях с использованием реальных измеренных параметров магнитов [2]. Для достижения проектных параметров пучков, параметры магнитного поля структурных элементов магнитной оптики должны с высокой точностью соответствовать расчетным. Для решения этих задач при серийном производстве магнитов выполняется измерение параметров магнитного поля структурных элементов магнитной оптики.

Задача измерения параметров магнитных элементов преследует две главные цели:

1) Убедиться, что параметры магнитного поля изготовленного магнита соответствует заданным допускам;
2) Получить необходимое и достаточное количество параметров, позволяющих моделировать движение пучка в вакуумной камере магнита.

Параметры магнитного поля измеряемых магнитов получаются в результате применения процедур обработки данных, записываемых при проведении измерений с магнитометрических датчиков. После анализа полученных результатов делается заключение о соответствии параметров магнита расчетным.

Информацию во время испытаний или моделирования следует собирать правильным образом. Слишком часто данные хранятся без описательной информации, в несовместимых форматах и разбрасываются по большому числу компьютеров. Это создает массив информации, который чрезвычайно затрудняет поиск определенного набора данных и принятие решений на его основе. Тесты или симуляции приходится повторять, когда наборы данных не удается найти.

В результате этого снижается эффективность, и увеличиваются затраты на разработку.

Важной составляющей процесса получения параметров магнитов является организация хранилища: для «сырых» данных, результатов промежуточных вычислений и для окончательных наборов.

В настоящей работе речь пойдет о структуризации способа обработки и хранения результатов магнитных измерений.

Постановка задачи

Цель работы: Проектирование базы данных для результатов магнитных измерений.

Задачи:

1) Сбор информации:
   - 1.1 Подробно изучить текущую систему хранения;
   - 1.2 Изучить процесс обработки данных (результатов магнитных измерений);
   - 1.3 Проанализировать проблемы, характерные для текущих способов организации процессов и интеграции между ними;
   - 1.4 Оценить последствия выявленных недостатков и рассмотреть вопрос о целесообразности их модификации с целью улучшения.
2) Сформировать основные требования для проектирования структуры базы данных:
   - 2.1 Выделить условия, которым должно соответствовать хранилище данных;
   - 2.2 Оценить возможность улучшения и/или переиспользования существующих интеграционных решений для снижения стоимости разработки, внедрения и поддержки компонентов Базы Данных.
3) Осуществить модификацию выделенных процессов и сделать выбор инструментов на основе проанализированной информации в пунктах 1−2 :
   - 3.1 Провести сравнение инструментов, позволяющих реализовать все выделенные потребности. Сделать выбор в пользу наиболее подходящего.
   - 3.2 Предложить схему организации хранилища

В качестве критериев должны выступать:

- Достижение экономии ресурсов на поддержку и эксплуатацию полученного решения в сравнении с исходным вариантом;
- Возможность переиспользования данных для дальнейшего анализа.

4) Спроектировать систему отчетности:

   4.1 Создание решения визуализации для наблюдения и анализа результатов измерений.

5) Проанализировать полученные результаты;

6) Определить возможность распространения полученного решения на иные интеграционные сценарии. Оценить возможные затраты на переиспользование.

Глава 1. Анализ существующей системы хранения и обработки результатов магнитных измерений

Для того, чтобы грамотно организовать базу данных для результатов магнитных измерений, необходимо тщательно разобраться во всех этапах обработки данных – от получения первичных («сырых») данных с датчиков магнито–метрической системы (ММС) до конечного набора параметров, получаемых после программной обработки.

Основной предмет исследования: система обработки и хранения результатов магнитных измерений, направленных на получение параметров магнитов, которые серийно производит отдел магнитных измерений.

Задачи:

- Описать бизнес-процессы, на обеспечение выполнения которых направлено интеграционное решение;
- Описать и оценить исходную систему хранения и обработки данных. Выделить ключевые преимущества и недостатки;
- Предложить пути улучшения. Установить требования для создания базы данных.

1.1. Методология проведения магнитных измерений, основные принципы

Каждый магнит будущего ускорителя проходит процесс тщательной проверки на строгое соответствие допускам. Этот процесс заключается в проведении серийных испытаний на стенде магнитной лаборатории. В результате испытаний получают наборы параметров магнитного поля, значение каждого из которых должно попасть в строгий заранее установленный диапазон.

Измерения состоят в определении потока вектора индукции магнитного поля через катушку путем интегрирования наведенной в ней ЭДС по времени и повторяются пошагово с разным угловым положением катушек.

Чтобы не смещать фокус на физико-математические основы и принципы работы самой ММС, приведем только алгоритм сбора результатов измерений [6, 7, 8, 9].

На каждом шаге измерительного цикла выполняется следующая последовательность действий:

1) Генерируется управляющий источником питания сигнал, задающий форму и амплитуду импульса тока;
2) Оцифровываются соответствующие сигналы напряжений с гармонических катушек и сигнал с датчика тока;
3) Производится запись полученных данных на жесткий носитель;
4) При помощи сервопривода выполняется поворот гармонических катушек магнитометра на угол $\Delta\theta = 2\pi/N$ (N – количество измерений для фурье-анализа)

Цикл измерений повторяется три раза:

1) В реперном магнитном поле для определения азимутального расположения катушек относительно геометрической средней плоскости магнита;
2) В основном поле магнита в режиме «без компенсации сигнала» для определения дипольной компоненты A_{1i}, B_{1i};
3) В основном поле магнита с «компенсацией сигнала» для определения более высоких гармоник A_{ni}, B_{ni} (индекс i означает номер катушки).

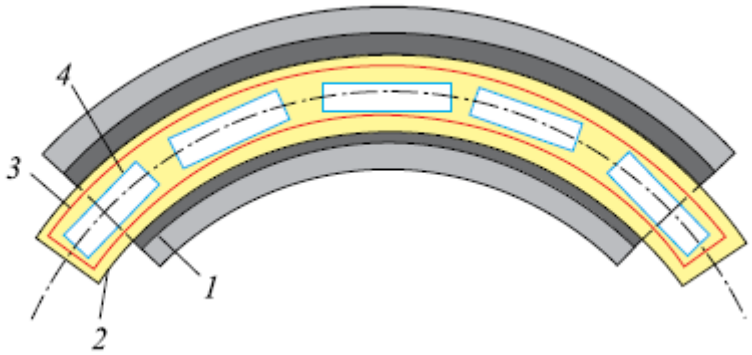

Рис. 2. Схема расположения измерительных стационарных катушек в дипольном магните бустера: 1 – ярмо магнита; 2 – ложемент датчика; 3 – виток для определения эффективной длины; 4 – измерительные катушки

Для каждого магнита проводятся «теплые» (при температуре окружающей среды) и «холодные» (при подключении системы охлаждения жидким гелием) измерения. Непосредственно каждый вид измерений повторяется 3-5 раз, чтобы гарантировать отсутствие ошибок в единичном измерении и получить повторяемость значений. В рамках проекта NICA встречаются два типа магнитов – дипольные и квадрупольные. У каждого типа магнитов своя методика измерений (из-за различия геометрической формы требуется различные измерительные установки (ММС)).

1.2. Схема процессов сбора данных и обработки результатов магнитных измерений

Теперь, когда мы получили ясное представление о процессе проведения магнитных измерений, можем составить более подробную картину процесса обработки данных.

В системе сбора данных используется оборудование National Instruments PXI (рис.3), в частности, сбор данных и управление питанием магнита осуществляется на базе шасси PXIe1065, а управление сервомотором с помощью cRIO-9031 − через стандарт промышленных сетей EtherCAT. Управление и сбор данных проводится в программной среде, созданной на языке программирования LabView. [6, 7, 8, 9].

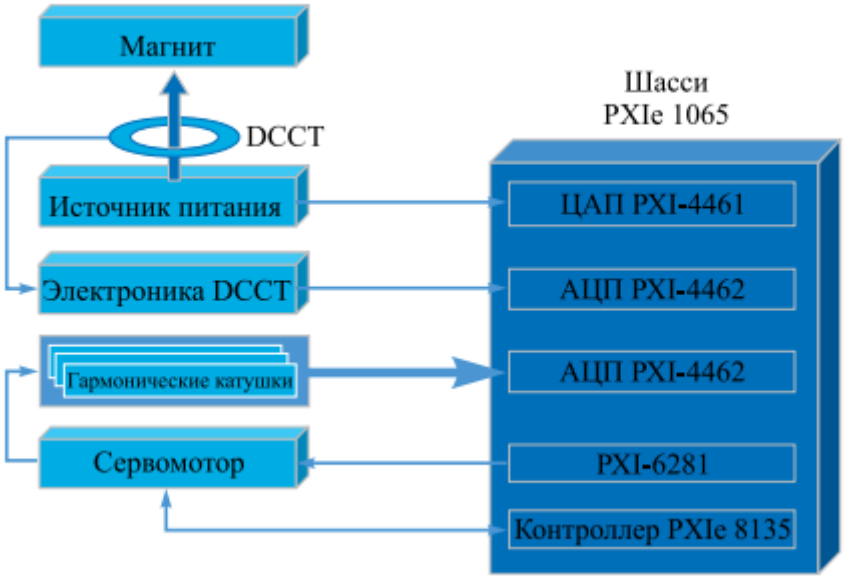

Рис. 3. Схема сбора данных и управления системой

Данные с измерительной установки претерпевают дальнейшую обработку:

- Результатами измерений являются TDMS-файлы с интегральными кривыми, записанными с датчиков сенсора − «сырые» данные;
- В среде LabView реализован алгоритм, который обрабатывает TDMS-файлы с данными для каждой серии измерений, выполняет математические вычисления (интегрирование, фурье-анализ) и формирует таблицы Программа выводит ряд таблиц с результатами параметров;
- Далее в Origin или Excel по таблицам разных серий вычисляют средние значения и отклонения, также формируется ряд таблиц с результатами искомых параметров;
- Полученные параметры сравнивают с расчетными;
- Делают вывод о соответствии магнита установленным параметрам допуска, а также об успешности самих вычислений;
- Если все результаты прошли контроль, то требуемые параметры (только часть из вычисленных) заносится в паспорт магнита −печатный заверенный протокол с характеристиками.

Рис. 4. Схема последующей обработки данных

1.3. Особенности форматов данных магнитных измерений

Особое внимание требуется уделить формату данных. Результатами измерений являются файлы в формате TDMS с интегральными кривыми, записанными с датчиков сенсора − «сырые» данные.

Файл TDMS−это файл данных, сохраненный в формате потоковой передачи управления техническими данными National Instruments (NI) [10]. Он содержит данные моделирования или измерения, записанные с помощью программного обеспечения National Instruments, такого как LabVIEW и DIAdem. Файлы TDMS также хранят описательную информацию, которая включает в себя процедуру, тестовое устройство, информацию о датчике и автора.

Формат TDMS был разработан National Instruments, чтобы ученым и инженерам было проще хранить большие объемы данных, записанных в результате испытаний и моделирования. Формат TDMS поддерживается различными программами National Instruments, включая LabVIEW и DIAdem. Он также поддерживается MathWorks MATLAB и OpenOffice Calc. Можно выполнять чтение в Excel.

Компания National Instruments определила решение для управления техническими данными (TDM), которое включает три неотъемлемых части:

1) Формат файла NI TDMS для сохранения хорошо документированных данных измерений
2) NI DataFinder для быстрого поиска ранее сохраненных наборов данных
3) NI DIAdem или тулкит LabVIEW DataFinder для обработки данных и создания отчетов

Обзор структуры файла TDMS

Единственная, самая важная особенность во внутреннем формате структуры файла TDMS, это ее внутренняя иерархическая организация. Формат файла TDMS упорядочен с использованием трех уровней иерархии (Рис.5). Уровень файлов может содержать неограниченное количество групп, и каждая группа может содержать неограниченное количество каналов. Благодаря такой группировке каналов появляется возможность выбирать, как организовать данные, чтобы их было легче понять. Например, в пределах одного файла может быть одна группа для исходных данных и другая группа для данных после анализа. Или может быть несколько групп, которые соответствуют типам датчиков или местоположению.

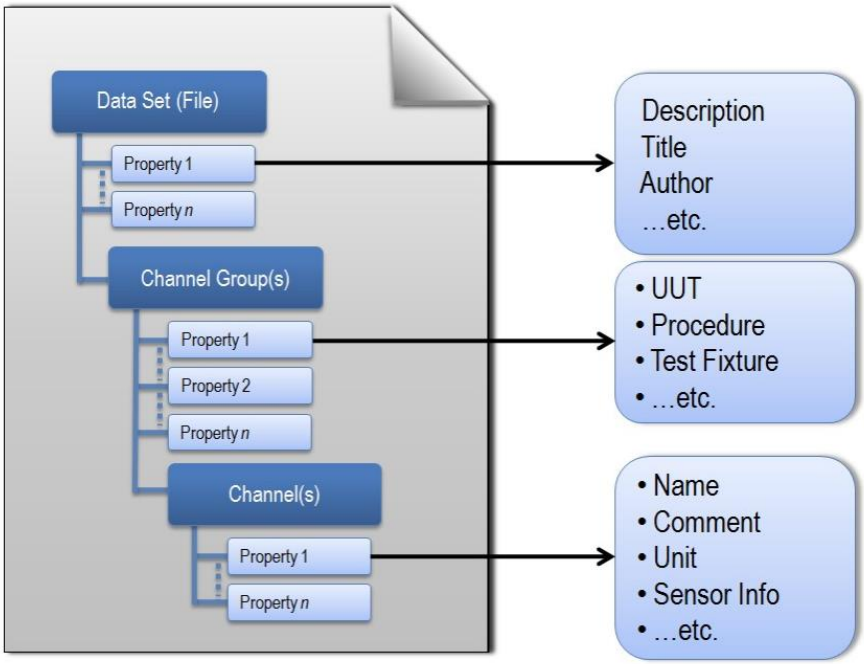

Рис. 5. Структура TDMS− файла содержит описательную информацию на уровнях файла, группы и канала.

Файлы TDMS_INDEX создаются автоматически с файлами TDMS. Индексные файлы содержат информацию о связанном файле TDMS, на который ссылается приложение, открывающее файл TDMS. Это

позволяет быстрее открывать файлы TDMS, в которых хранятся большие наборы данных.

Однако данные, преобразованные в формат TDMS, еще не содержат конечных параметров магнита. Они должны пройти еще два этапа вычислительной обработки, в результате которой будут получены искомые параметры.

1.4. Обзор получаемых параметров

Для магнитов коллайдера NICA созданы системы, позволяющие измерять требуемые согласно спецификации параметры с заданными точностями.

Измерения проводятся над двумя видами магнитов:

1) Дипольные;
2) Квадрупольные (фокусирующие/дефокусирующие).

Измерительные установки существенно отличаются друг от друга в связи с различием геометрических форм магнитов.

Однако, есть общие наборы конечных параметров для всех магнитов:

- B_1 – значение поля в центре магнита (или градиента G для квадрупольной линзы)
- L_{eff} – эффективная длина магнита
- $\Delta\theta$ – угол наклона медианной плоскости
- $a_1 - a_{10}$, $b_1 - b_{10}$ – интегральные величины относительных гармоник магнитного поля до десятой включительно

Для квадрупольных линз еще измеряют координаты:

- $\Delta X, \Delta Y$ – координаты оси

Все эти параметры рассчитывают как для «теплых», так и для «холодных» измерений.

В рамках данной работы нет необходимости приводить расчетные формулы и более подробное описание особенностей различных ММС, т.к. цель работы спроектировать базу данных для результатов измерений. Нам следует четко представлять наборы данных на всех этапах обработки, чтобы наиболее грамотно организовать схему базы данных.

1.5. Актуализация потребности в организации хранилища. Перспективы.

Существующая система организации обработки и хранения данных имеет как ряд преимуществ, так и недостатков. Рассмотрим наиболее подробно все аспекты.

Преимущества:

- Для хранения первичных результатов измерений используются зеркальные RAID-массивы;
- Формат файлов TDMS очень удобен для структуризации результатов измерений;
- Есть решение на LabView, которое обрабатывает TDMS-файлы с данными для каждой серии измерений и формирует таблицы со значениями параметров.

Недостатки:

- Нет единого структурированного хранилища для основных результатов измерений − таблиц с набором полученных после обработки параметров;
- Протоколы / паспорта магнитов хранятся в разрозненном виде на серверах лаборатории, однако есть распечатанные заверенные бумажные версии документов;
- Нет возможности проводить полноценную аналитику, наблюдать статистику в целом по всем данным;
- При возникновении потребности сформировать отчет по результатам измерений, приходится вручную находить нужные файлы с таблицами, выбирать нужные для отчета поля также вручную, дополнительно посчитать средние значения и

отклонения в Excel, Origin или написать скрипт на LabView, и в программе Word сформировать отчет с комментариями;

- Повторная обработка данных затруднена, нет удобной организации процесса;
- Данные доступны только тем специалистам, которые проводят непосредственно сами измерения, в то время как эти данные могут также потребоваться заказчику или другим коллегам. Необходимо организовать возможность удаленного доступа для тех, кому будет предоставлено разрешение, для дальнейшего анализа данных.

Перспективы:

На данном этапе реализации проекта текущее решение вполне оправдывает свое существование − серийных измерений над магнитами проведено еще не так много, сейчас испытания проводят только для магнитов бустера (предускорителя) или для магнитов от заказчиков из CERN-а, однако данных становится все больше, а вскоре потребуется проводить измерения над магнитами основного кольца коллайдера, которых существенно больше, чем для бустера.

Совершенно точно можно сказать, что оставаться при текущей системе хранения данных по меньшей мере ненадежно – в силу следующих возможных проблем:

- Предстоит серьезное увеличение объемов данных на несколько порядков;
- Проблема медленного поиска;
- Вероятность ошибки при поиске и расчетах вручную гораздо выше;
- Отсутствие удаленного доступа к данным;

- Нет удобного подхода, чтобы полноценно анализировать данные и вести статистику наблюдений;
- Не автоматизирована система отчетности.

1.6. Основные требования к структуре базы данных

Условия, которым должно соответствовать хранилище данных:

1) Структурированное архивное хранение и поиск на основе запросов;
2) Горизонтальная масштабируемость;
3) Параллельная обработка данных;
4) Отказоустойчивость;
5) Аналитические возможности;
6) Возможность автоматизации системы отчетности;
7) Возможность удаленного доступа к данным.

Оценим возможность улучшения и/или переиспользования существующих интеграционных решений для снижения стоимости разработки, внедрения и поддержки компонентов Базы Данных:

- В процессе обработки магнитных измерений надежно работает программа обработки данных в среде LabView, которая использует сложные математические операции, такие как интегрирование, ряды Фурье и т.д. − разумным решением будет сохранить этот этап, т.к. в современных СУБД процесс реализации данных вычислений будет очень затратным как с точки зрения времени, так и с точки зрения стоимости реализации.
- Сырые данные надежно хранятся в формате TDMS, зеркальные RAID-массивы обеспечивают защиту от потери данных. Все коннекторы настроены на этот формат, и существующее решение не имеет существенных недостатков кроме одного − этот формат нельзя напрямую (без преобразования) загрузить в Базу Данных. Вероятно, на первом этапе разработки мы не будем изменять эту часть системы, т.к. она исправно работает и удобна для ученых.

- Мы обратимся к результатам обработки файлов TDMS, и в первую очередь, организуем новую структурированную систему хранения именно для них.

Глава 2. Проектирование Базы Данных

Современное программное обеспечение крупных физических экспериментов основывается на наборе различного рода программных сред, компонентов и модулей. При разработке программного модуля, входящего в состав инфраструктуры эксперимента, следует учесть существование большого числа различного рода данных и соответствующих программных служб. Например, технические данные, данные геометрии установки, базы данных системы сбора данных и системы контроля физической установки. Также существуют данные событий, условий и конфигураций. Используются различные службы управления данными.

Первым шагом к созданию единого решения для управления данными является обеспечение того, чтобы данные хранились наиболее эффективным, упорядоченным и расширяемым образом, а также реализация рационального доступа к данным.

Задачи:

- На основе проведенного анализа выбрать подходящую СУБД;
- Разработать структуру хранилища;
- Осуществить и описать процесс реализации.

2.1 Общие принципы организации хранилища для экспериментальных данных

Особенности программного обеспечения в области крупных экспериментов заключаются в следующем:

- Большое число данных, подлежащих регистрации, хранению, обработке и визуализации;
- Широкий ряд программных систем предназначен для выполнения обработки данных в реальном времени.

В основе систем хранения экспериментальных данных как правило лежат стандартные РСУБД (Oracle, MySQL и т.д.). На основе стандартных РСУБД могут разрабатываться надстройки, ориентирующие систему хранения на экспериментальные данные. Также в рамках СУБД реализуются оптимальные и эффективные схемы организации хранения данных (таблицы, представления, хранимые процедуры, хранимые типы и т.д.). В ряде случаев и специфических задач используются файловые системы хранения на основе технологии XML и файлов уникального типа.

В области экспериментальной физики используются два широких подхода, к хранению данных: хранение данных на основе использования файлов и использования реляционных баз данных. Оба подхода хранения дополняют друг друга и используются совместно. Файловое хранение обычно используется для больших наборов экспериментальных данных.

Хранилища на основе баз данных используется, когда требуется распределенная обработка дынных, обычно с централизованной «записью» данных: и распределенным чтением, когда требуется индексация для выполнения быстрых запросов на умеренных объемах

данных, и где требуется структурированная архивное хранение и поиск на основе запросов, а также одновременная запись многими пользователями; и поддержка транзакций.

Конкретная реализация систем хранения экспериментальных данных не тривиальна и решаются в рамках отдельного эксперимента. Обычно на основе реляционных СУБД создаются системы хранения, при этом внутренняя структура реализована в виде множества связанных таблиц, реализующих как функции непосредственного хранения данных, так и механизмы рационального доступа и поиска. Также в реляционных базах данных хранятся ссылки на внешние файлы данных.

Экспериментальные данные событий поступают с большой частотой и в больших объемах. При этом первично полученные данные позднее пройдут несколько этапов автономной обработки. Наиболее рациональным способом хранения; таких данных является запись их в большие файлы необработанных данных с уникальными именами, которые впоследствии будут обработаны. При этом соответствующие условия, при которых был получен такой файл, а именно данные использованных конфигураций-рабочих систем и т.д., записываются в реляционные базы данных.

Также реляционная база данных хранит ссылки на файлы данных, соответствующие каждому набору данных условий, таким образом, каталогизируя файлы. В качестве указателя внешнего файла данных может выступать, например, номер экспериментального запуска. Таким образом, система хранения данных современных экспериментов представляет собой комбинированное хранение на основе файловых и реляционных баз данных. Такая организация позволяет эффективно записывать большой объем данных событий, при этом одновременная

поддержка реляционных баз данных позволяет эффективно выполнять запросы и получать необходимые файлы данных.

Хранилище данных является лучшей в своем роде платформой для работы со структурированными данными [12]. Его архитектура обеспечивает более высокую производительность и обходится гораздо дешевле любых других альтернатив, в том числе таких популярных платформ, как Hadoop и Apache Spark, которые лишь дополняют и расширяют хранилище данных. Они не подходят для аналитической обработки запросов, что как раз и является основной функцией хранилища данных.

Многое изменилось с тех пор, как 30 лет назад была впервые представлена архитектура хранилища данных. На смену строчным базам данных с непараллельной обработкой, которые лежали в основе традиционных решений, пришли гораздо более быстрые и эффективные колоночные платформы, где хранение данных организовано по столбцам и используется массивно-параллельная обработка (MPP). Новшеством является не сама MPP-обработка, а ее доступность.

Не так давно аналитические базы данных сделали MPP привлекательной с экономической точки зрения, дав возможность выполнять масштабирование с разумными затратами. Для подавляющего числа запросов хранилище данных MPP обеспечивает очень высокую производительность — несопоставимо больше, чем реляционные СУБД без параллельной обработки. При выполнении аналитических запросов колоночное хранилище данных с массивно-параллельной обработкой работает еще быстрее.

2.2 Обоснование выбора колоночной СУБД

Архитектура традиционных хранилищ данных разрабатывалась на основе строчных реляционных СУБД, рассчитанных на обработку транзакций в реальном времени (OLTP), таких как Oracle, DB2 и Sybase.

Не так давно выбор строчных СУБД в качестве основы для хранилищ данных объяснялся тем, что в то время это было лучшее, самое удобное, экономичное и мобильное решение. Но развитие колоночных баз данных изменило ситуацию [13].

В отличие от задач OLTP, в аналитические запросы включается лишь ряд атрибутов, а это означает, что для основной части аналитики колоночная архитектура обеспечит максимальную производительность и эффективность[14].

Выбор колоночной СУБД оправдывает непревзойденная скорость работы. В традиционных реляционных СУБД информация хранится в строках, поэтому, даже если вашему запросу нужны данные из одного столбца, СУБД сканирует содержимое всех строк — каждого столбца в каждой строке. Таким образом, с диска каждый раз считывается вся строка.

Применяемый сегодня подход создает большую нагрузку на системы ввода-вывода и увеличивает время поиска данных, т.к. для ответа на один простой запрос требуется обработать очень много данных.

Второе главное преимущество колоночного хранилища заключается в том, что его архитектура позволяет сократить объем операций ввода-вывода данных — а это самый существенный параметр для аналитической обработки.

Третье преимущество — колоночные СУБД эффективно сжимают данные, благодаря чему обеспечивают в 4–5 раз более высокую производительность, чем традиционные СУБД.

Строчные базы данных устарели, колоночная платформа значительно ускорит анализ данных, обеспечив и ряд других важнейших преимуществ.

К тому же строчные реляционные СУБД не способны эффективно сжимать данные. Мало того, практика показывает, что строчные СУБД, объединенные в ходе создания сервиса хранилища данных, могут использоваться вместе лишь при наличии программного средства поддержки принятия решений, которое будет обеспечивать совместную работу с ними. При таком подходе настройка и балансировка становятся более трудоемкими процедурами, чем при использовании колоночных СУБД, причем это становится особенно заметно по мере дальнейшего увеличения объемов данных. Кроме того, для обеспечения нужной производительности администраторам СУБД приходится реализовывать временные решения: материализованные представления, вертикальное партиционирование и индексацию каждого столбца.

В результате возрастают расходы на эксплуатацию и обслуживание хранилища данных. И чем больше объемы данных, многообразнее и сложнее запросы к базе данных, тем ниже эффективность их выполнения.

Если в базе данных имеются сотни или тысячи столбцов, то традиционная СУБД будет работать очень медленно, т.к. все их нужно считывать. Чтобы ускорить обработку данных, приходится реализовывать одно или несколько обходных решений, таких как материализованные представления или индексы. Эти обходные решения имеют ограничения. Материализованные представления необходимо

обновлять и поддерживать, индексы предварительно строить и т. д. В конечном счете именно от этого и зависит их эффективность.

Нужно заметить, что нет ни одной технологии, которая решила бы все проблемы.

Но недостатки колоночного хранилища превращаются в достоинства, когда речь идет не о транзакционной нагрузке, характерной для учетных систем, а об аналитической нагрузке. В частности, одна из особенностей колоночной платформы заключается в том, что колоночная СУБД хранит каждый столбец в отдельном множестве файлов.

Колоночная СУБД, как и ее реляционный аналог, оптимизируется для физического хранения данных на диске или в оперативной памяти. Это важное различие между СУБД, например колоночной или РСУБД OLTP, и хранилищем данных, например Hadoop.

Развертывание колоночной СУБД в ее базовой конфигурации может привести к замедлению операций добавления и обновления данных, а также задержать или усложнить их загрузку.

Чтобы избежать подобных проблем, в колоночных СУБД, в частности в Vertica, реализуются оптимизированные технологии массивно-параллельной загрузки и используются реляционные подходы OLAP (ROLAP) или комплексные методы OLAP (MOLAP). А также колоночное проектирование позволяет применять и другие технологии оптимизации, повышающие производительность, например блочную итерацию. Так, несколько значений из столбца могут быть переданы от одного оператору другому в виде блока, что оказывается более эффективным, чем использование итераторов в строчной схеме.

Сейчас некоторые поставщики «встраивают» в строчные базы данные возможности колоночного хранилища. Например, база данных

Microsoft SQL Server 2016 и более поздние версии поддерживают «колоночное индексирование», при этом ее модуль расширения PowerPivot Excel позволяет на обычных ПК использовать функции, аналогичные колоночному хранению. Но эти продукты не только не реализуют колоночную архитектуру СУБД, но и не обеспечивают массивно-параллельную обработку.

2.3 Выбор Базы Данных. Сравнение инструментов

Основываясь на выборе колоночной базы данных, проведем сравнительный анализ доступных вариантов [15].

Name	**EXASOL** X	**Greenplum** X	**Microsoft SQL Server** X	**Oracle** X	**Vertica** X
Description	High-performance, in-memory, MPP database specifically designed for in-memory analytics.	Analytic Database platform built on PostgreSQL. Full name is Pivotal Greenplum Database	Microsofts relational DBMS	Widely used RDBMS	Columnar relational DBMS designed to handle modern analytic workloads, enabling fast query performance
Primary database model	Relational DBMS	Relational DBMS	Relational DBMS	Relational DBMS	Relational DBMS
Secondary database models		Document store	Document store Graph DBMS	Document store Graph DBMS RDF store	Document store
Server-side scripts	user defined functions	yes	Transact SQL, .NET languages, R, Python and (with SQL Server 2019) Java	PL/SQL	yes
Triggers	yes	yes	yes	yes	no
Partitioning methods	Sharding	Sharding	tables can be distributed across several files (horizontal partitioning); sharding through federation	horizontal partitioning	Sharding
Replication methods		Master-slave replication	yes, but depending on the SQL-Server Edition	Master-master replication Master-slave replication	Master-master replication
MapReduce	yes	yes	no	no	yes
Consistency concepts	Immediate Consistency	Immediate Consistency	Immediate Consistency	Immediate Consistency	Immediate Consistency
Foreign keys	yes	yes	yes	yes	yes
Transaction concepts	ACID	ACID	ACID	ACID	ACID
Concurrency	yes	yes	yes	yes	yes
Durability	yes	yes	yes	yes	yes
In-memory capabilities	yes	no	yes	yes	
User concepts	Access rights for users, groups and roles according to SQL-standard	fine grained access rights according to SQL-standard	fine grained access rights according to SQL-standard	fine grained access rights according to SQL-standard	fine grained access rights according to SQL-standard; supports Kerberos, LDAP, Ident and hash
XML support	no	yes	yes	yes	yes
Secondary indexes	yes	yes	yes	yes	no
SQL	yes	yes	yes	yes	yes
APIs and other access methods	.Net JDBC ODBC WebSocket	JDBC ODBC	ADO.NET JDBC ODBC OLE DB Tabular Data Stream (TDS)	JDBC ODBC ODP.NET Oracle Call Interface (OCI)	ADO.NET JDBC Kafka ODBC Proprietary protocol RESTful HTTP API
Supported programming languages	Java Lua Python R	C Java Perl Python R	C# C++ Delphi Go Java JavaScript (Node.js) PHP Python R Ruby Visual Basic	C C# C++ Clojure Cobol Delphi Eiffel Erlang Fortran Groovy Haskell Java JavaScript Lisp Objective C OCaml Perl PHP Python R Ruby Scala Tcl Visual Basic	C++ Java Perl Python R

Рис.6. Таблица для сравнения колоночных СУБД

Предусматривая возможность загрузки в хранилище конфиденциальных данных, мы не рассматриваем облачные решения.

Приведенные для сравнения колоночные СУБД призваны решить проблему неэффективной работы традиционных СУБД в аналитических системах и системах в подавляющим большинством операций типа «чтение».

Анализируя данные по совокупности параметров различных колоночных решений, выделим наиболее оптимальное.

- Oracle и Microsoft SQL Server не имеют возможности горизонтального масштабирования. Это решение нам не подходит, т.к. предполагается существенное увеличение объемов данных и требуется гибкое решение;
- EXASOL требует установки дорогостоящих серверов, сложна в администрировании;
- Greenplum масштабируется только вертикально, что потенциально может стать проблемой («бутылочное горлышко»);
- Методом исключения (а также удовлетворения поставленным условиям) наиболее оптимальным вариантом становится Vertica.

Она обладает интуитивно понятным интерфейсом, управлять данными сможет любой специалист, владеющий базовыми знаниями языка SQL, а также Vertica предоставляет наиболее комфортную возможность горизонтального масштабирования – на недорогом и маломощном оборудовании получить прирост скорости выполнения запросов в 5–10, а иногда даже в 100 раз, при этом, благодаря компрессии, данные будут занимать на диске в 5–10 раз меньше места, чем в случае с традиционными СУБД.

Еще раз отметим, что не существует универсального решения. Ни одна платформа не может идеально справиться абсолютно со всеми аналитическими задачами. Однако мы можем выделить наиболее подходящую для наших условий.

2.4 Vertica

Vertica – платформа для анализа больших данных (Big data) в реальном времени [16]. Vertica не просто хранит свои данные в колонках, она делает это рационально, с высокой степенью сжатия, а также эффективно планирует запросы и быстро отдает данные. Информация, которая в классической строчной СУБД занимает около 1 ТБ дискового пространства, на Vertica займет порядка 200-300 Гб, тем самым мы получаем хорошую экономию на дисках.

В частности, Vertica, анализируя данные, автоматически подбирает наиболее оптимальный алгоритм сжатия из имеющихся в ее распоряжении. Кроме того, колоночная архитектура Vertica позволяет выполнять позднюю материализацию — осуществлять операции со сжатыми столбцами в оперативной памяти.

Vertica изначально проектировалась именно как колоночная СУБД. Другие базы в основном стараются имитировать различные колоночные механизмы, но не всегда удачно, т.к. движок все равно создан чтобы обрабатывать строки. Как правило, идея состоит в том, чтобы транспонировать таблицу и далее обрабатывать ее привычным строчным механизмом.

Vertica стала одной из первых колоночных с массивно-параллельной обработкой систем хранилищ данных. Она также имеет преимущество колоночной MPP с точки зрения производительности.

Можно достичь увеличения производительности до 1000% по скорости выполнения запросов. Подобное увеличение производительности означает, что операции, которые занимали один час, вполне смогут выполниться за 6 минут, а обработку, которая длится в течение восьми часов, потребуется всего 48 минут.

Vertica — это в первую очередь аналитическое хранилище данных. Её следует рассматривать как некий batch-слой, куда стоит загружать данные большими порциями. При необходимости Vertica очень быстро готова эти данные отдать — запросы на миллионы строк выполняются за секунды.

Vertica отказоустойчива, в ней нет управляющей ноды — все ноды равны. Если с одним из серверов в кластере возникают проблемы, то данные все равно будут получены.

Рассмотрим Vertica на уровне кластера. Эта СУБД обеспечивает массивно-параллельную обработку данных (MPP) в распределенной вычислительной архитектуре — «shared-nothing» — где, в принципе, любая нода готова подхватить функции любой другой ноды. Основные свойства:

- отсутствует единая точка отказа;
- каждый узел независим и самостоятелен;
- отсутствует единая для всей системы точка подключения;
- узлы инфраструктуры дублируются;
- данные на узлах кластера автоматически копируются.

Кластер удобно линейно масштабируется. Достаточно добавить серверы и подключить их через графический интерфейс. Помимо серийных серверов, возможно развертывание на виртуальные машины.

С помощью расширения можно добиться:

- увеличения объема для новых данных;
- увеличения максимальной рабочей нагрузки;
- повышения отказоустойчивости.

Чем больше нод в кластере, тем меньше вероятность выхода кластера из строя из-за отказа, а следовательно, возможно обеспечение доступности 24/7.

При удвоении количества серверов в кластере хранилища с массивно-параллельной обработкой производительность возрастает в два раза. Удобно, что ценообразование в отношении Vertica, аналитической СУБД с MPP потерабайтное, что позволяет масштабировать производительность хранилища с учетом потребностей бизнеса. Это очень важное отличие от традиционных систем, которые лицензируются по числу процессоров, количеству пользователей, подключенных систем или по всем этим параметрам сразу. Есть возможность бесплатного использования с ограничением в 1 ТБ.

Работа с Hadoop

Т.к. хранение в виде файловой системы актуально для экспериментальных данных, рассмотрим в качестве примера возможность работы с платформой Hadoop (представляет open-source решение, наиболее подходящее хранения как структурированных, так и неструктурированных данных)[17].

Hadoop также является средой с массивно-параллельной обработкой данных, однако, в отличие от реляционных СУБД, Hadoop не является СУБД, она реализует распределенную файловую систему. Hadoop (HDFS) обеспечивает хранение и управление данными на множестве серверов (узлов), объединенных в кластер.

Реляционные СУБД с MPP, например Vertica, тоже распределяют данные по кластеру. Основное различие между реляционной СУБД и файловой системой (которую все-таки можно считать в каком-то смысле базой данных) — это уровень абстракции. В большинстве случаев СУБД реализует оптимизацию размещения и физической структуры, логику и схему работы с данными «поверх» файловой системы. HDFS не обладает необходимыми для этого функциями. Как и реляционная СУБД, Hadoop может эффективно хранить и извлекать данные, которые

записывает в блоки одинакового размера, но в Hadoop нет встроенных возможностей для сопоставления данных, хранящихся в этих блоках.

Если Hadoop использовать как платформу для выполнения SQL-запросов, то, чтобы воспользоваться функциями, присущими реляционной СУБД, придется внедрить технологии Hive, Impala, Drill или Spark (со Spark SQL). Однако эти инструменты не предлагают возможностей, необходимых для управления данными. В отличие от Vertica, которая совместима с версией стандарта ANSI SQL, а платформы Hive, Impala и Spark SQL не полностью совместимы с ANSI SQL, к тому же не поддерживают высокий уровень параллелизма.

Ядро Vertica способно работать с аналогом колоночного хранилища, а именно с файлами Parquet и Optimized Row Columnar (ORC), которые используются с Impala и Hive. С помощью системы обработки SQL-запросов можно получить прямой доступ к данным в Hadoop (например, файлам Parquet или ORC) и проанализировать их. Таким образом, Vertica является очень гибким вариантом для хранения данных.

Однако, чтобы работать с Hadoop напрямую, необходимо заниматься обучением имеющихся специалистов, искать новых сотрудников и приобретать дополнительное ПО, а для этого потребуется трудозатрат – как временных, так и материальных. С переходом на Vertica появляется возможность отказаться от дорогостоящих хранилищ данных. Это способ модернизировать существующее решение без больших потрясений.

2.5 Архитектура хранилища данных

На основе проведенного анализа можно сделать выбор в пользу традиционной архитектуры хранилища данных.

В отличие от традиционной SQL-СУБД, Data Warehouse (DWH) имеет сложную многоуровневую (слоеную) архитектуру, которая называется LSA — Layered Scalable Architecture. LSA реализует логическое деление структур с данными на несколько функциональных уровней. Данные копируются с уровня на уровень и трансформируются при этом, чтобы в итоге предстать в виде согласованной информации, пригодной для анализа [12].

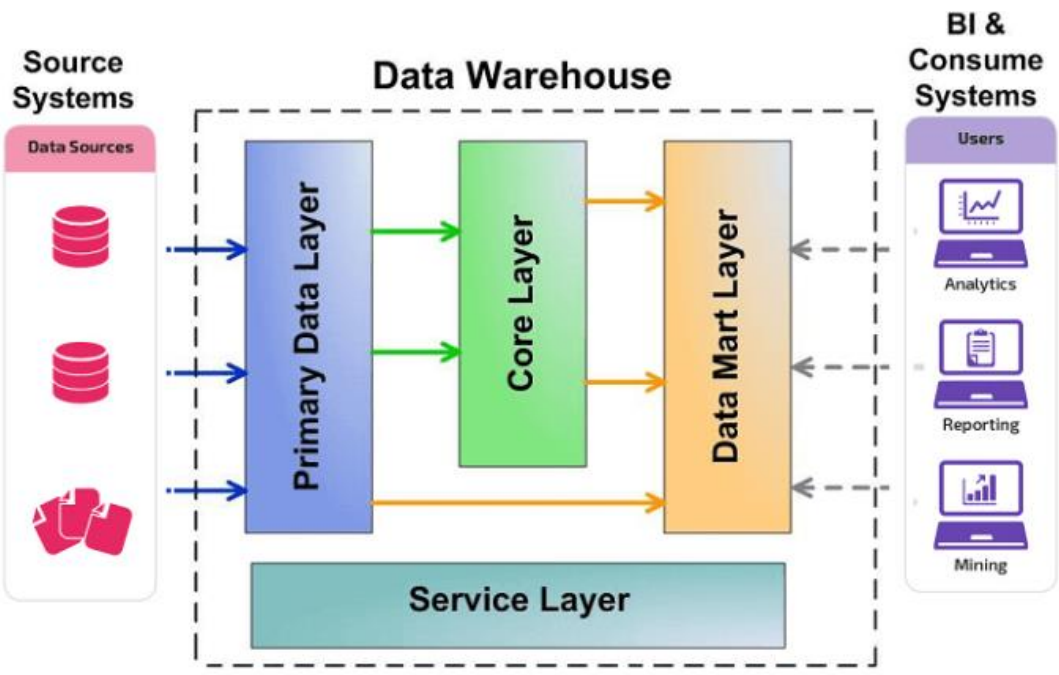

Рис. 7. слоеная архитектура DWH: как устроено хранилище данных

Такая архитектура включает в себя три главных уровня:

1) Операционный слой первичных данных (Primary Data Layer или Staging) – уровень, на котором выполняется загрузка информации из систем-источников в исходном качестве с сохранением полной истории изменений. Здесь происходит абстрагирование следующих слоев хранилища от физического устройства источников данных, способов их сбора и методов выделения изменений.
2) Ядро хранилища (Core Data Layer) или Detailed Data Store (DDS) – центральный компонент, который выполняет консолидацию данных из разных источников, приводя их к единым структурам и ключам. Именно здесь происходит основная работа с качеством данных и общие трансформации, чтобы абстрагировать потребителей от особенностей логического устройства источников данных и необходимости их взаимного сопоставления. Так решается задача обеспечения целостности и качества данных.
3) Витрина данных (Data Mart) – уровень, где данные преобразуются к структурам, удобным для анализа и использования в BI-дашбордах или других системах-потребителях. Используется для отчетности и анализа конкретных бизнес-линий. В этой модели хранилища – агрегированные данные из ряда исходных систем. Витрина данных позволяет увидеть агрегированную информацию в определенном временном или тематическом разрезе, а также сформировать и распечатать отчетные данные в виде шаблонизированного документа.

Таким образом, распределим наши данные по уровням:

1) Primary Data Layer: Первичные результаты измерений, преобразованные из формата TDMS в формат Parquet (это бинарный, колоночно-ориентированный формат хранения больших данных, позволяющий использовать преимущества сжатого и эффективного колоночно-ориентированного представления информации и возможность загрузки в Vertica)
2) Core Data Layer: сформированные в таблицы результаты обработки измерений
3) Data Mart: агрегированные данные для протоколов, отчетов, с реализацией бизнес-логики.

2.6 Схема Базы Данных для слоя CDL

Рассмотрим организацию результатов магнитных измерений на уровне ядра хранилища (Рис. 7. Core Data Layer)

Исходя из того, что получаемые наборы параметров почти одинаковы для всех видов магнитов, можем объединить их в одну большую таблицу – таблицу фактов.

Остальные имеющиеся данные содержат описательную информацию к проведенным измерениям − данные об измерительной установке, магнито-метрической системе, параметрах катушек, типе измерений, температуре, коэффициентах и т.д. Их можно распределить по нескольким таблицам измерений.

Таким образом, схема типа «звезда» наилучшим образом подходит для организации такой структуры данных. Она имеет централизованное хранилище данных, которое хранится в таблице фактов. Таблица фактов содержит значения всех параметров всех серийных измерений, а таблицы измерений описывают хранимые данные.

Сейчас для имеющихся данных вполне достаточно двух таблиц измерений:

1) для общих данных об измеряемом магните (id, имя и номер магнита, тип, ярмо, обмотка, геометрические размеры и т.д.);
2) для данных о проведенном измерении (id, название, дата, тип измерений, температура, тип ММС, параметры катушек и т.д.).

Предлагаемая схема базы данных представлена на Рис. 8.

Денормализованные проекты менее сложны, потому что данные сгруппированы. Таблица фактов использует только одну ссылку для присоединения к каждой таблице измерений.

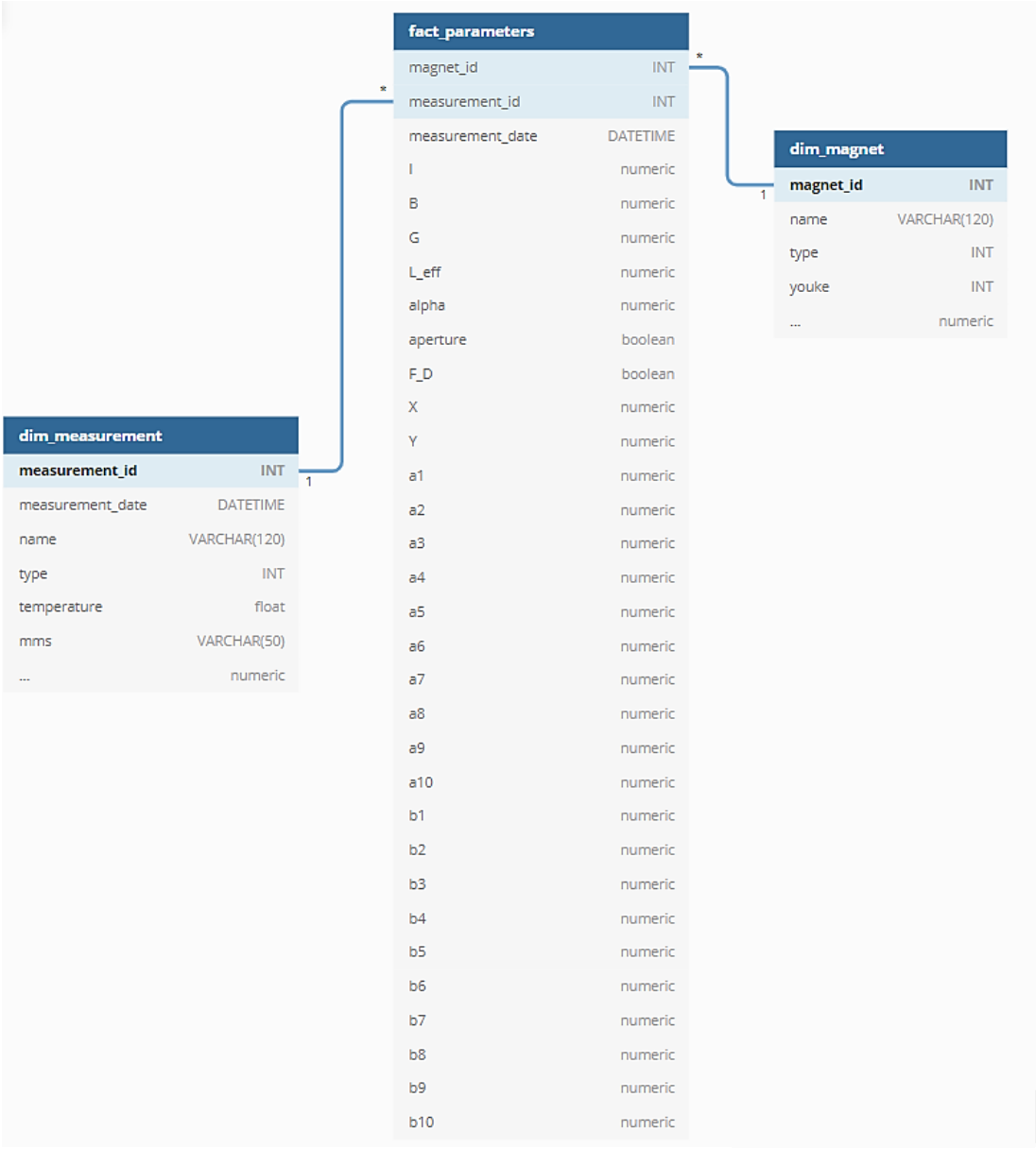

Рис. 8. Схема базы данных

В дальнейшем количество таблиц измерений можно увеличить, разбив уже имеющиеся на более мелкие для нормализации, или добавив совершенно новые.

2.7 Витрина данных

Витрина содержит агрегированные данные из основного слоя данных, предназначенные для аналитики, построения графиков, протоколов, отчетов, с реализацией бизнес-логики.

На этом уровне базы данных будут реализованы:

- подсчет средних значений и отклонений выбранных параметров;
- выборка нужных параметров для сравнения с ТЗ;
- автоматизированное сравнение полученных параметров с расчетными (останется лишь сформировать и распечатать отчетные данные из шаблона документа).

Витрины данных имеют следующие достоинства:

- пользователи видят и работают только с теми данными, которые им нужны;
- витрины являются максимально приближенными к пользователю;
- витрины данных проще в проектировании, настройке и поддержке, чем полноразмерное хранилище;
- для витрин данных не требуется использовать мощные вычислительные средства.

К недостаткам витрин данных можно отнести сложность контроля целостности и противоречивости данных.

2.8 Визуализация данных

Для построения графиков, реализации бизнес-логики и представления результатов измерений в виде дашбордов, доступных для любых пользователей, предлагается использовать Tableau Server.

Ключевые преимущества:

1) Универсальность: Tableau позволяет подключаться к любым источникам данных, включая файлы, реляционные СУБД, аналитические кубы и «облачные источники», такие как Google BigQuery. Есть возможность формировать обоснованные и точные отчеты разного назначения независимо от сложности аналитических задач.
2) Гибкость: Удобный конструктор для создания индивидуальных отчетов и визуализаций. Любой объем данных будет организован в соответствии с конкретными потребностями компании и приобретет удобную форму. Можно подключаться к данным и строить отчеты без помощи IT-специалистов.
3) Удобство и скорость: Tableau предлагает пользователю интуитивно понятный интерфейс, гибкость в решении сложных вопросов, возможность разнообразно визуализировать данные на разных этапах работы с ними.

При подключении Tableau к витрине данных можно организовать удобный интерфейс для визуализации данных для любых пользователей, а также настроить автоматизированную систему отчетности.

Возможны и более простые инструменты. Например, Microsoft Excel.

Глава 3. ELT- процесс

Важной частью процесса организации хранилища является процесс загрузки данных в хранилище. Необходим удобный инструмент для автоматизации процессов экспорта, загрузки и трансформации данных.

Задачи:

- Предложить инструмент для управления ELT−процессами;
- Описать реализацию процесса загрузки данных в хранилище.

3.1 Apache Airflow

Airflow [18] — это инструмент для разработки, планирования и мониторинга процессов загрузки и обработки данных[9]. Основная особенность Airflow в том, что для разработки процессов используется язык Python.

Рассмотрим основные сущности Airflow. Их суть и назначение — оптимальная организация архитектуры процессов. Основной сущностью в Airflow является Directed Acyclic Graph (DAG).

DAG (направленный ацикличный граф) — это некоторое смысловое объединение задач, которые необходимо выполнить в строго определенной последовательности по определенному расписанию.

Airflow представляет удобный web-интерфейс для работы с графами и другими сущностями. Проектируя граф, закладывается набор вершин (операторов), с помощью которых будут построены задачи внутри графа.

Оператор — это сущность, на основании которой создаются экземпляры заданий, где описывается, что будет происходить во время

исполнения экземпляра задания. Например, наиболее часто используют операторы:

- SqlOperator — для выполнения SQL-кода;
- PythonOperator — для вызова Python-кода;
- BashOperator — для выполнения bash-команды;
- Sensor — ожидания события. Например, нужного времени, появления необходимого файла, обновления таблицы в базе данных;
- EmailOperator — для отправки email;
- HTTPOperator — для работы с http-запросами.

Кроме того, можно разрабатывать собственные операторы, ориентируясь на особенности, и использовать их в проекте. По сути, как только в проекте возникает часто используемый код, построенный на базовых операторах, можно задуматься о том, чтобы собрать его в новый оператор. Это упростит дальнейшую разработку и пополнит собственную библиотеку операторов в проекте.

Все созданные экземпляры задач выполняет планировщик. Планировщик задач в Airflow построен на Celery [19]. Celery — это Python-библиотека, позволяющая организовать очередь с асинхронным и распределенным исполнением задач. Со стороны Airflow все задачи делятся на пулы. Пулы создаются вручную. Обычно, их целью является ограничение нагрузки на работу с источником или типизировать задачи внутри DWH. Пулами можно управлять через web-интерфейс.

3.2 Реализация графов в Airflow

Для реализации процессов загрузки и обработки данных в хранилище (Рис.7) достаточно построить несколько типов направленных ациклических графов (DAG).

1) Граф, который проверяет готовность данных и загружает TDMS-файлы в слой Primary Data Layer (PDL);
2) Граф, который проверяет готовность данных и загружает результаты обработки магнитных измерений в слой Core Data Layer (CDL);
3) Граф, который проверяет готовность данных и загружает результаты обработки магнитных измерений в слой DataMart.

Таким образом все алгоритмы могут работать параллельно.

Выводы

1) Проведен сравнительный анализ и осуществлен обоснованный выбор распределённой колоночной СУБД Vertica;
2) Спроектирована архитектура хранилища для результатов магнитных измерений проекта NICA;
3) С помощью технологии Apache Airflow реализован процесс запуска алгоритмов загрузки данных в хранилище.

Спроектированное хранилище удовлетворяет всем поставленным требованиям и предусматривает возникновение новых потребностей в будущем.

Заключение

В рамках данной работы велась разработка базы данных для хранения и обработки результатов магнитных измерений.

Основной целью при разработке структуры базы данных для проекта NICA являлось создание универсального решения, реализующего систему хранения и обработки результатов магнитных измерений, представляющего также удобный интуитивно−понятный интерфейс для доступа к данным, хранящимся в СУБД.

Выбор СУБД Vertica обусловлен простотой администрирования базы данных и удовлетворением всем установленным требованиям. Для доступа к данным системы могут быть использованы запросы SQL. Преимуществами этого выбора также являются возможности массивно-параллельной обработки данных, горизонтальной масштабируемости, высокой скорости обработки данных, а также одним из важнейших свойств является возможность качественного сжатия большого количества данных.

Архитектура хранилища представляет из себя трёхуровневую структуру:

1) Primary Data Layer: Первичный слой данных, содержит первичные результаты измерений;
2) Core Data Layer: детализированный слой данных, содержит сформированные в таблицы результаты обработки измерений;
3) Data Mart: витрина данных, которая содержит агрегированные данные для протоколов, отчетов, с реализацией бизнес-логики.

Инструмент Apache Airflow позволил реализовать программные методы автоматического управления загрузкой данных в хранилище, а также обеспечила интерфейсный программный слой между

внутренними уровнями организации хранилища и доступ к данным для средств графической визуализации. С его помощью построены направленные ацикличные графы, которые позволяют параллельно запускать алгоритмы, проверяя, что необходимые данные доступны в хранилище.

Для визуализации предложен инструмент Tableau, который предоставляет удобный пользовательский интерфейс, и напрямую подключается к витрине данных. С помощью Airflow можно автоматизировать систему отчетности.

Разнообразие используемых средств обуславливает проблему согласования форматов данных.

Предложенное решение реализовано в локальном окружении.

В задачи проекта не входила задача замены ранее использовавшихся программ обработки данных. Удалось успешно адаптировать существующее решение и внедрить его в хранилище данных.

В будущем предполагается интеграция уже существующих модулей обработки данных к единой программной шине и достижения автоматизации всех этапов обработки целиком внутри хранилища.

Список литературы

1. Kekelidze V. et al. NICA Project at JINR // Письма в ЭЧАЯ. 2012. Т. 9, No.4/5. P.521-526.
2. Андрианов С. Н. Динамическое моделирование систем управления пучками частиц. // Издательство СПбГУ. СПб, 2002.
3. Bryant P. J. Basic theory for magnetic measurements // CERN Accelerator School, Montreux, Switzerland, CERN 92-05,1992. P.52–69.
4. Khodzhibagiyan H. et al. 2014. Superconducting Magnets for the NICA Accelerator-Collider Complex pp. 4001304, June. IEEE Trans.Appl.Supercond., 24, N3 (параметры дипольного магнита).
5. Gourber J. P. Philosophy of series measurements // CERN Accelerator School. Montreux. Switzerland. CERN 92-05, 1992. P. 84–102.
6. Borisov V. et al. Magnetic measurement system for the NICA booster magnets // Proc. 5-th Intern. Particle Accelerator Conf. – IPAC 2014. Dresden, Germany. June 15– 20, 2014. P. 2696–2698.
7. Костромин С.А., Борисов В.В., Бычков А.В., Донягин А.М. и др. Измерение характеристик магнитного поля дипольного магнита бустера NICA // Письма в ЭЧАЯ. 2016. Т.13, №7(205). С.1333−1342.
8. Шандов М.М., Акишин П.Г., Борисов В.В., Бычков А.В., Донягин А.М., Костромин С.А. и др. Магнитные измерения предсерийных двухапретурных дипольных магнитов коллайдера NICA // Письма в ЭЧАЯ . 2018. Т. 15, №7(219). С. 832−844.
9. Shevchuk A., Borisov V., Bychkov A., Donyagin A., Kostromin S. at al. Serial magnetic measurements of quadrupole magnets of the NICA booster synchrotron // Письма в ЭЧАЯ. 2018. Т. 15, №7(219). С. 845.
10. TDMS−format Documentation [Электронный ресурс]. URL: https://www.ni.com/ru-ru/support/documentation/supplemental/06/the-ni-tdms-file-format.html (дата обращения: 3.04.20).

11. Коннолли Т., Бегг К. Database Systems: A Practical Approach to Design, Implementation, and Management. // 3-е изд. 2003. P.1436
12. Patil, Preeti S.; Srikantha Rao; Suryakant B. Patil. Optimization of Data Warehousing System: Simplification in Reporting and Analysis. // IJCA Proceedings on International Conference and Workshop on Emerging Trends in Technology (ICWET). Foundation of Computer Science. P. 33–37.
13. S. Harizopoulos; D. Abadi; P. Boncz. Column-Oriented Database Systems. // VLDB 2009 Tutorial. p. 5.
14. D. J. Abadi; S. R. Madden; N. Hachem (2008). Column-stores vs. row-stores: how different are they really? // SIGMOD'08. pp. 967–980.
15. System Properties Comparison [Электронный ресурс]. URL: https://dbengines.com/en/system/ htm (дата обращения: 5.04.20).
16. Vertica Documentation [Электронный ресурс]. URL: https://my.vertica.com/docs/9.2.x/HTML/index.htm (дата обращения: 7.04.20).
17. Vertica Documentation [Электронный ресурс]. URL: https://www.vertica.com/wpcontent/uploads/2019/04/Vertica_DWH_modernisation_ru.pdf (дата обращения: 7.04.20).
18. Koppen V., Bruggemann B., Berendt B. Designing Data Integration: The ETL Pattern Approach // Cepis Upgrade. 2011. Vol. 13. P. 49–55.
19. Apache Airflow Documentation [Электронный ресурс]. URL: https://airflow.apache.org/project.html/ (дата обращения: 28.04.20).