Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

*К защите допустить:*

И.О. Заведующего кафедрой информатики

\_\_\_\_\_\_\_\_\_\_\_\_\_С. И. Сиротко

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

**РЕАЛИЗАЦИЯ ПРОСТОЙ БАЗЫ ДАННЫХ. ПРОГРАММНЫЙ МОДУЛЬ, ПЕРЕНОСИМЫЙ БЕЗ ИНСТАЛЛЯЦИИ. ХРАНЕНИЕ ДАННЫХ СРЕДСТВАМИ ФАЙЛОВОЙ СИСТЕМЫ С ДОСТУПОМ ПОСРЕДСТВОМ УПРОЩЕННОГО ЯЗЫКА ЗАПРОСОВ.**

БГУИР КП 1-40 04 01 020 ПЗ

Студент А. В. Скворцов

Руководитель Н. Ю. Гриценко

Нормоконтролер Н. Ю. Гриценко

Минск 2024

**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 3](#_Toc162304615)

[**1 БАЗЫ ДАННЫХ** 4](#_Toc162304616)

[1.1 Общая информация 4](#_Toc162304617)

[1.2 Классификация баз данных 7](#_Toc162304618)

[**2 ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ** 12](#_Toc162304619)

[**3 СТРУКТУРНЫЙ ЯЗЫК ЗАПРОСОВ** 12](#_Toc162304620)

[**4 ПРОЕКТИРОВАНИЕ ФУНКЦИОНАЛЬНЫХ ВОЗМОЖНОСТЕЙ ПРОГРАММЫ** 12](#_Toc162304621)

[**5 ВЗАИМОДЕЙСТВИЕ С РАЗРАБОТАННОЙ БАЗОЙ ДАННЫХ** 12](#_Toc162304622)

# **ВВЕДЕНИЕ**

Технология баз данных появилась почти полвека назад и с тех пор не только оказала огромное влияние на развитие информационных технологий, но и кардинально изменила методы работы многих организаций и предприятий. В наше время сложно найти такую компанию, в которой не использовались бы базы данных, точнее, информационные системы, основанные на базах данных. Возросший поток информации приводит к необходимости разработки новых приемов ее осмысления и обработки. Накопленный опыт и развитие программного обеспечения позволяет переосмыслить такую традиционную область обработки информации как хранение и управление данными. Новый подход к организации процесса обработки информации приводит к понятию база данных и методам работы с ней.

Целью данного курсового проекта является разработка программного модуля, который позволит реализовать простую базу данных с хранением данных средствами файловой системы и обеспечит доступ к данным через упрощенный язык запросов. Основными задачами работы являются создание программного модуля, который будет обеспечивать функциональность базы данных. Модуль должен быть переносимым, то есть его можно использовать на различных платформах без необходимости установки дополнительных компонентов. Так же в базе данных будет разработан механизм хранения данных, используя файловую систему компьютера. Будет рассмотрен язык запросов, который применяют, чтобы работать с базами данных, структурированных особым образом, и будет представлена его упрощенная реализация для управления разработанным модулем. Язык будет являться легким для понимания и использования, а также будет поддерживать основные операции для работы с данными, такие как добавление, обновление, удаление и выборка данных.

# **БАЗЫ ДАННЫХ**

* 1. Общая информация

Высокая ценность информации признавалась во все времена, но только во второй половине XX в., когда появилась возможность эффективного управления действительно большими объемами информации, она стала важнейшим стратегическим ресурсом, обслуживание которого потребовало создания специализированных программно-технических средств — автоматизированных информационных систем (АИС).

АИС обеспечивают надежное хранение и оперативное обновление информации, а также ее поиск, извлечение и аналитическую обработку по запросам потребителей. При всем разнообразии архитектур, решаемых задач и условий использования АИС в структуре их программного обеспечения (ПО) явно выделяют два относительно автономных компонента: подсистему хранения данных и подсистему обработки информации. Основу подсистемы хранения составляют база данных (БД) — программно-реализованная информационная модель предметной области АИС, управляемая специализированной программной системой (СУБД), и подсистема обработки информации, включающая множество прикладных программ, получающих доступ к СУБД для чтения и/или модификации информации, хранимой в базе данных.

АИС относятся к категории социотехнических систем, важнейшим компонентом которых, наряду с аппаратным и программным обеспечением, являются пользователи — разработчики, администраторы и конечные пользователи, участвующие в создании или эксплуатации АИС на различных стадиях ее жизненного цикла.

Различают АИС, основанные на знаниях, и АИС, основанные на данных. К первым можно отнести, например, экспертные системы (ЭС), интеллектуальные системы поддержки принятия решений (СППР) и т.п. Ко вторым – всевозможные прикладные системы, которые сейчас активно используются и на предприятиях, и в учреждениях. Такие прикладные системы применяются очень широко.

Существуют две основные предпосылки создания таких систем:

1. Разработка методов конструирования и эксплуатации систем, предназначенных для коллективного использования.
2. Возможность собирать, хранить и обрабатывать большое количество данных о реальных объектах и явлениях, то есть оснащение этих систем "памятью".

Массив данных общего пользования в системах, основанных на данных, называется базой данных. База данных (БД) является моделью предметной области информационной системы.

На заре развития вычислительной техники обрабатываемые данные являлись частью программ: они располагались сразу за кодом программы в так называемом сегменте данных (рисунок 1.1, а). Следующим шагом стало хранение данных в отдельных файлах (рисунок 1.1, б). Недостатком этих двух подходов являлась зависимость программ от данных: сведения о структуре данных включались в код программы. При изменении структуры данных необходимо было вносить изменения в программу.

Логичным продолжением этой эволюции является перенос описания данных в массив данных (рисунок 1.1, в). Это позволило обеспечить независимость данных от программ.

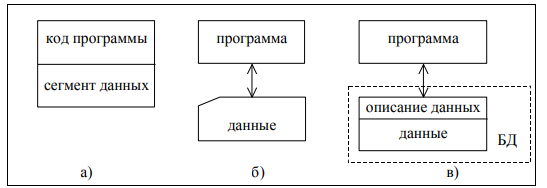


Рисунок 1.1 – Развитие принципов разработки баз данных

***База данных*** – это организованная коллекция данных, хранящихся и обрабатываемых в компьютерной системе. Она предназначена для эффективного хранения, управления и извлечения информации. Базы данных используются в различных областях, включая бизнес, науку, образование и государственное управление.

Базы данных позволяют эффективно хранить большие объемы данных, обеспечивать структурированный доступ к информации и обрабатывать данные с помощью запросов и операций. Они позволяют пользователям добавлять, изменять, удалять и извлекать данные в удобном формате. Базы данных также поддерживают функции безопасности, целостности данных и резервного копирования, чтобы обеспечить сохранность и доступность информации.

***Система управления базами данных*** (СУБД) – это комплекс языковых и программных средств, предназначенный для создания, ведения и совместного использования базой данных многими пользователями.

В процессе исследований, посвященных тому, как именно должна быть устроена СУБД, американским комитетом по стандартизации ANSI (American National Standards Institute) сформулирована трехуровневая система организации БД (рисунок 1.2).

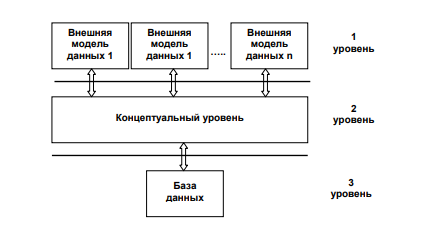


Рисунок 1.2 – Трехуровневая модель системы управления базами данных

1. *Уровень внешних моделей* – самый верхний уровень, где каждая модель имеет свое «видение» данных. Этот уровень определяет точку зрения на БД отдельных приложений. Каждое приложение видит и обрабатывает только те данные, которые необходимы именно этому приложению.
2. *Концептуальный уровень* – центральное управляющее звено, здесь БД представлена в наиболее общем виде, который объединяет данные, используемые всеми приложениями, работающими с данной БД. Фактически концептуальный уровень отражает обобщенную модель предметной области, для которой создавалась БД.
3. *Физический уровень* – собственно данные, расположенные в файлах или в страничных структурах, расположенных на внешних носителях информации. Эта архитектура позволяет обеспечить логическую (между уровнями 1 и 2) и физическую (между уровнями 2 и 3) независимость при работе с данными. Первая предполагает возможность изменения одного приложения без корректировки других приложений, работающих с этой же базой данных. Физическая независимость предполагает возможность переноса хранимой информации с одних носителей на другие при сохранении работоспособности всех приложений, работающих с данной базой данных.
   1. Классификация баз данных

Важнейшим достоинством применения баз данных является обеспечение независимости данных от прикладных программ. Это даст возможность пользователям не заниматься проблемами представления данных на физическом уровне: размещения данных в памяти, методов доступа к ним и т.д. Такая независимость достигается поддерживаемым системами управления базами данных многоуровневым представлением данных в базе на логическом (пользовательском) и физическом уровнях. Благодаря системам управления базами данных и наличию логического уровня представления данных обеспечивается отделение концептуальной модели базы данных от ее физического представления в памяти ЭВМ.

Логическую структуру хранимых в базе данных называют моделью представления данных. К основным моделям представления данных относятся следующие*: реляционная, иерархическая, сетевая, объектно-ориентированная.* Некоторые системы управления базами данных могут одновременно поддерживать несколько моделей данных.

**Реляционная модель** является простейшей и наиболее привычной формой представления данных в виде таблицы. В теории множеств для нее имеется развитый математический аппарат – реляционное исчисление и реляционная алгебра. Достоинством реляционной модели является сравнительная простота инструментальных средств ее поддержки, недостатком – жесткость структуры данных (невозможность, например, задания строк таблицы произвольной длины) и зависимость скорости ее работы от размера базы данных.

Реляционная модель данных предложена сотрудником фирмы IBM Эдгаром Коддом и основывается на понятии отношение (relation). Отношение представляет собой множество элементов, называемых кортежами.

Наглядной формой представления отношения является привычная для человеческого восприятия двумерная таблица. Таблица имеет строки (записи) и столбцы (поля). Каждая строка таблицы имеет одинаковую структуру и состоит из полей. Строкам таблицы соответствуют кортежи, а столбцам – атрибуты отношения. С помощью одной таблицы удобно описывать простейший вид связей между данными, а именно деление одного объекта (явления, сущности, системы и проч.), информация о котором хранится в таблице, на множество подобъектов, каждому из которых соответствует строка или запись таблицы. При этом каждый из подобъектов имеет одинаковую структуру или свойства, описываемые соответствующими значениями полей записей. Например, таблица может содержать сведения о группе обучаемых, о каждом из которых известны следующие характеристики: фамилия, имя и отчество, пол, возраст и образование. Поскольку в рамках одной таблицы не удается описать более сложные логические структуры данных из предметной области, применяют связывание таблиц.

Физическое размещение данных в реляционных базах на внешних носителях легко осуществляется с помощью обычных файлов.

Достоинство реляционной модели данных заключается в простоте, понятности и удобстве физической реализации на ЭВМ. Именно простота и понятность для пользователя явились основной причиной их широкого использования. Проблемы эффективности обработки данных этого типа также оказались технически вполне разрешимыми.

Основными недостатками реляционной модели является отсутствие стандартных средств идентификации отдельных записей и сложность описания иерархических и сетевых связей.

**Постреляционная модель** данных представляет собой расширенную реляционную модель, снимающую ограничение неделимости данных, хранящихся в записях таблиц.

Постреляционная модель данных допускает многозначные поля - поля, значения которых состоят из подзначений. Набор значений многозначных полей считается самостоятельной таблицей, встроенной в основную таблицу.

Достоинством постреляционной модели является возможность представления совокупности связанных реляционных таблиц одной постреляционной таблицей. Это обеспечивает высокую наглядность представления информации и повышение эффективности ее обработки.

Недостатком постреляционной модели является сложность решения проблемы обеспечения целостности и непротиворечивости хранимых данных. Для описания структуры иерархической модели используется тип данных – «дерево». Тип «дерево» является составным и включает в себя подтипы («поддеревья»), каждый из которых, в свою очередь, является типом «дерево». Каждый из типов «дерево» состоит из одного «корневого» типа и упорядоченного набора подчиненных типов (рисунок 1.3).

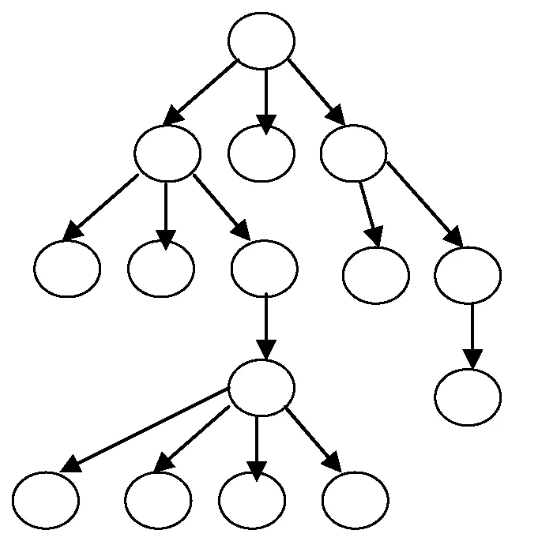


Рисунок 1.3 – Иерархическая модель данных

Корневым называется тип, который имеет подчиненные типы и сам не является подтипом. Подчиненный тип (подтип) является потомком по отношению к типу, который выступает для него в роли предка (родителя). Потомки одного и того же типа являются близнецами по отношению друг к другу. Каждый из элементарных типов, включенных в тип «дерево», является простым или составным типом «запись». Поля записей хранят собственно числовые или символьные значения, составляющие основное содержание базы данных.

Обход всех элементов иерархической базы данных обычно производится сверху вниз и слева направо. Иерархическая модель предполагает наличие связей между данными, имеющими какой-либо общий признак.

В иерархической модели такие связи могут быть отражены в виде дерева-графа, где возможны только односторонние связи от старших вершин к младшим. Это облегчает доступ к необходимой информации, но только если все возможные запросы отражены в структуре дерева. Никакие иные запросы удовлетворены быть не могут.

К достоинствам иерархической модели данных относятся эффективное использование памяти ЭВМ и неплохие показатели времени выполнения основных операций над данными. Иерархическая модель данных удобна для работы с иерархически упорядоченной информацией. Недостатком иерархической модели является ее громоздкость для обработки информации с достаточно сложными логическими связями, а также сложность понимания для обычного пользователя.

Сетевая модель данных позволяет отображать разнообразные взаимосвязи элементов данных в виде произвольного графа, обобщая тем самым иерархическую модель. Для описания схемы сетевой базы данных используется две группы типов: «запись» и «связь». Тип «связь» определяется для двух типов «запись»: предка и потомка. Переменные типа «связь» являются экземплярами связей (рисунок 1.4).

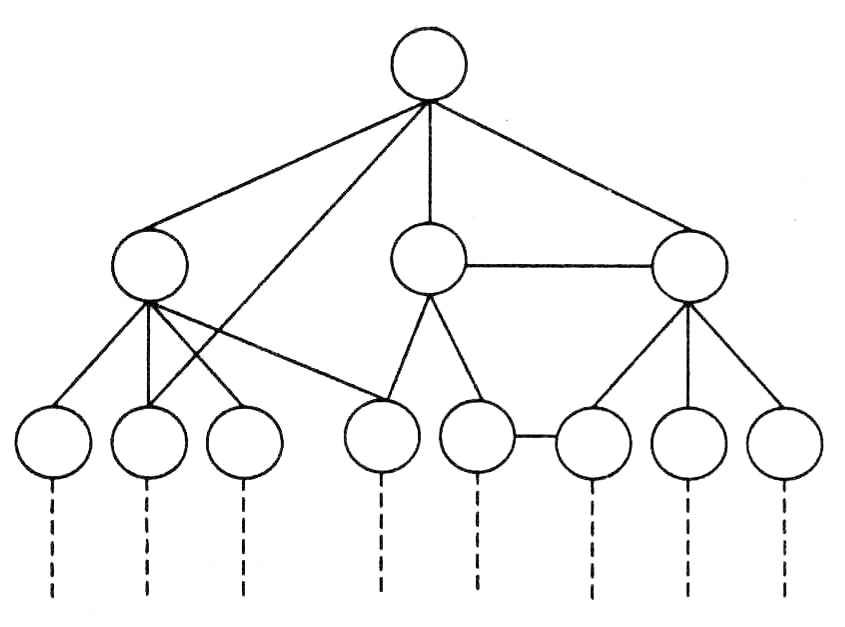


Рисунок 1.4 – Иерархическая модель данных

Сетевая БД состоит из набора записей и набора соответствующих связей. На формирование связи особых ограничений не накладывается. Если в иерархических структурах запись-потомок могла иметь только одну запись-предка, то в сетевой модели данных запись-потомок может иметь произвольное число записей-предков (сводных родителей).

Достоинством сетевой модели данных является возможность эффективной реализации по показателям затрат памяти и оперативности. В сравнении с иерархической моделью сетевая модель предоставляет большие возможности в смысле допустимости образования произвольных связей. Недостатком сетевой модели данных является высокая сложность и жесткость схемы БД, построенной на ее основе, а также сложность для понимания и выполнения обработки информации в БД обычным пользователем. Кроме того, в сетевой модели данных ослаблен контроль целостности связей вследствие допустимости установления произвольных связей между записями.

В объектно-ориентированной модели при представлении данных имеется возможность идентифицировать отдельные записи базы. Между записями базы данных и функциями их обработки устанавливаются взаимосвязи с помощью механизмов, подобных соответствующим средствам в объектно-ориентированных языках программирования.

Для выполнения действий над данными в рассматриваемой модели БД применяются логические операции, механизмы инкапсуляции, наследования и полиморфизма. Инкапсуляция ограничивает область видимости имени свойства пределами того объекта, в котором оно определено. Смысл такого свойства будет определяться тем объектом, в который оно инкапсулировано. Наследование, наоборот, распространяет область видимости свойства на всех потомков объекта. Если необходимо расширить действие механизма наследования на объекты, не являющиеся непосредственными родственниками (например, между двумя потомками одного родителя), то в их общем предке определяется абстрактное свойство. Полиморфизм означает способность одного и того же программного кода работать с разнотипными данными. Другими словами, он означает допустимость в объектах разных типов иметь методы с одинаковыми именами.

Во время выполнения объектной программы одни и те же методы оперируют с разными объектами в зависимости от типа аргумента. Основным достоинством объектно-ориентированной модели данных в сравнении с реляционной является возможность отображения информации о сложных взаимосвязях объектов. Объектно-ориентированная модель данных позволяет идентифицировать отдельную запись базы данных и определять функции их обработки. Недостатками объектно-ориентированной модели являются высокая понятийная сложность, неудобство обработки данных и низкая скорость выполнения запросов.

* 1. …

# **ПЛАТФОРМА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ**

* 1. Linux

Linux – это свободная и открытая операционная система, которая является платформой программного обеспечения с огромным разнообразием приложений и инструментов. Она была создана Линусом Торвальдсом в 1991 году и с тех пор стала одной из самых популярных и широко используемых операционных систем в мире.

Первоначально Линус Торвальдс не хотел продавать свою разработку. И не хотел, чтобы её продавал кто-то другой. Это было чётко прописано в уведомлении об авторских правах, помещённом в файл COPYING самой первой версии — 0.01. Причём требование Линуса налагало значительно более жёсткие ограничения на распространение Linux, чем те, которые провозглашались в лицензии GNU: не разрешалось взимать никаких денег за передачу или использование Linux. Но уже в феврале 1992-го года к нему стали обращаться за разрешением брать плату за распространение дискет с Linux, чтобы покрыть временные затраты и стоимость дискет. Кроме того, необходимо было считаться и с тем, что при создании Linux использовалось множество свободно распространяемых по интернету инструментов, самым важным из которых был компилятор GCC.

Авторские права на него оговорены в общественной лицензии GPL, которую изобрёл Ричард Столлман. Торвальдсу пришлось пересмотреть свое заявление об авторских правах, и, начиная с версии 0.12, он тоже перешёл на использование лицензии GPL.

С технической точки зрения, Linux представляет собой только ядро Unix-подобной операционной системы, отвечающее за взаимодействие с аппаратной частью компьютера и выполнение таких задач, как распределение памяти, выделение процессорного времени различным программам и так далее.

Кроме ядра, операционная система включает в себя множество различных утилит, которые служат для организации взаимодействия пользователя с системой. Успех Linux как операционной системы во многом обусловлен тем, что к 1991-му году в рамках проекта GNU уже было разработано множество утилит, свободно распространяемых в интернете. Проекту GNU не хватало ядра, а ядро, скорее всего, осталось бы невостребованным, если бы отсутствовали необходимые для работы утилиты. Линус Торвальдс оказался со своей разработкой в нужном месте в нужное время. И Ричард Столлман прав, когда настаивает на том, что операционную систему следует называть не Linux, а GNU/Linux. Но название Linux исторически закрепилось за этой ОС.

Linux является модульной и гибкой платформой, которая поддерживает различные архитектуры и аппаратное обеспечение. Он может работать на серверах, настольных компьютерах, мобильных устройствах, встроенных системах и даже на суперкомпьютерах. Благодаря этой универсальности Linux нашел применение в различных областях, включая веб-серверы, базы данных, научные исследования, разработку программного обеспечения, мультимедиа, мобильные устройства и многое другое.

Linux имеет мощную командную строку, которая предоставляет пользователю множество инструментов для управления системой и выполнения задач. Кроме того, существует множество графических сред разработки (IDE) и интерфейсов пользователя, которые делают работу с Linux более удобной для начинающих пользователей. Linux также поддерживает большое количество программного обеспечения. Существует обширный выбор приложений для различных задач, включая офисные пакеты, браузеры, мультимедийные инструменты, графические редакторы, разработчикам программного обеспечения и многое другое. Большинство из них доступны бесплатно и имеют открытый исходный код.

В целом, Linux представляет собой мощную и гибкую платформу программного обеспечения, которая обеспечивает большую свободу и контроль над компьютерной системой. Благодаря своей открытой природе и активному сообществу разработчиков, Linux продолжает развиваться и оставаться одним из главных игроков в мире операционных систем.

* 1. Устройство диска в Linux

Логический диск в Linux представляет собой раздел или том на физическом диске. Разделы позволяют разделить физический диск на несколько изолированных областей, каждая из которых может иметь свою файловую систему. Логические диски и их разделы представлены в виде файлов в каталоге /dev.

Для работы с логическими дисками в Linux используются утилиты, такие как *fdisk*, *parted* и *lsblk*. *fdisk* и *parted* позволяют создавать, изменять и удалять разделы на физическом диске, а *lsblk* выводит информацию о физических и логических дисках, а также о разделах.

Логические диски могут быть отформатированы с использованием различных файловых систем, таких как ext4, XFS, Btrfs и другие. Форматирование диска создает файловую систему, позволяющую хранить файлы и каталоги. На рисунке 2.1 приведена схема типичного диска в Linux.

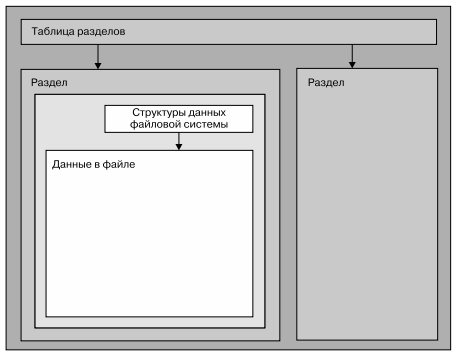


Рисунок 2.1 – Схема диска в Linux

Разделы являются более мелкими частями всего диска. В Linux они обозначаются с помощью цифры после названия блочного устройства и, следовательно, получают такие имена, как, например, /dev/sda1 и /dev/sdb3. Ядро представляет каждый раздел в виде блочного устройства, как если бы это был целый диск. Разделы определяются в небольшой области диска, которая называется таблицей разделов.

Хотя ядро и позволяет иметь одновременный доступ ко всему диску и к одному из его разделов, этого не придется делать, если только нет цели скопировать весь диск. Как можно заметить на рисунке 2.1, если пользователю необходим доступ к данным в файле, ему потребуется выяснить из таблицы разделов расположение соответствующего раздела, а затем отыскать в базе данных файловой системы этого раздела желаемый файл с данными.

Чтобы обращаться к данным на диске, ядро Linux использует систему слоев, показанную на рисунке 2.2. Подсистема SCSI и все связанное с ней представлены в виде одного контейнера. Следует обратить внимание на то, что с дисками можно работать как с помощью файловой системы, так и непосредственно через дисковые устройства.

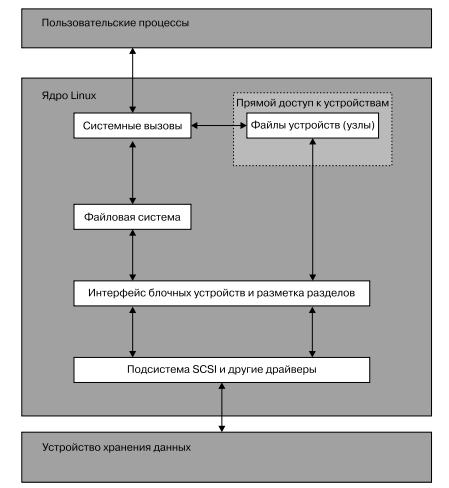
****

Рисунок 2.2 – Схема доступа ядра к диску

Существуют различные типы таблиц разделов. Традиционная таблица – та, которая расположена внутри главной загрузочной записи MBR (Master Boot Record). Новым, набирающим силу стандартом является глобальная таблица разделов с уникальными идентификаторами GPT (Globally Unique Identifier Partition Table).

Просмотр таблиц разделов – операция сравнительно простая и безвредная. Изменение таблиц разделов также осуществляется довольно просто, однако при таком типе изменений диска могут возникнуть опасности:

1. Изменение таблицы разделов сильно усложняет восстановление любых данных в удаляемых разделах, поскольку при этом меняется начальная точка привязки файловой системы. Обязательно нужно создавать резервную копию диска, на котором меняются разделы, если он содержит важную информацию.
2. Следует убедится в том, что на целевом диске ни один из разделов в данный момент не используется. Это важно, поскольку в большинстве версий Linux автоматически монтируется любая обнаруженная файловая система.
   1. Файловая система

Последним звеном между ядром и пространством пользователя для дисков обычно является файловая система. С ней помогают взаимодействовать, такие команды, как *ls* и *cd*. Как отмечалось ранее, файловая система является разновидностью базы данных; она поддерживает структуру, призванную трансформировать простое блочное устройство в замысловатую иерархию файлов и подкаталогов, которую пользователи способны понять