1 ОБЩАЯ ЧАСТЬ

* 1. Актуальность проекта

Документированная информация составляет основу управления, его эффективность в значительной степени базируется на производстве и потреблении информации. В современном обществе информация стала полноценным ресурсом производства, важным элементом социальной и политической жизни общества. Качество информации определяет качество управления. В современных условиях для повышения качества управления необходимо уделять достаточное внимание совершенствованию работы с документами, так как управленческое решение всегда базируется на информации, носителем которой является документ на различных основах.

В последнее время все больше предприятий сталкиваются с проблемой улучшения управляемости компании: улучшение контроля и ускорение бизнес-процессов, улучшение возможности их отслеживания, оптимизация рабочего времени, экономия трудозатрат, повышение производительности труда и так далее. Единственным способом реализации подобных задач является внедрение автоматизированной системы. Однако в настоящее время попытка решения задачи внедрения такой системы наталкивается на существенные сложности.

В современной организации руководитель сталкивается с необходимостью решать множество задач в условиях дефицита рабочего времени. Среди основных источников проблем в работе современного менеджера можно выделить следующие:

Большое количество процессов, в которых задействован руководитель, исполняющий при этом разные роли: инициатора, ответственного исполнителя, контролера.

Огромное количество документов, которые требуют согласования, уточнения, реакции, отслеживания, исполнения.

Необходимость контролировать своевременное исполнение заданий и отслеживание взятых на себя обязательств.

Постоянно растущие объемы дополнительной информации, необходимой для принятия управленческих решений.

С каждым днем все большее значение приобретает оперативность в принятии решений, контроль исполнения решений, рациональное использование накопленной информации и предыдущего опыта.

Тема автоматизации процессов документооборота становится в последнее время все более и более актуальной. Все больше компаний достигает необходимого уровня зрелости, и приходит к необходимости формализации бизнес-процессов и внедрения средств для их автоматизации. С другой стороны к настоящему моменту компании прошли длительный путь по созданию своих информационных систем, что обеспечивает потенциальные возможности для реализации задач подобного уровня сложности. Однако менеджеры информационных систем сталкиваются с большими проблемами определении методики и выборе программного обеспечения, которое способно обеспечить адекватные средства для реализации задач данного класса.

В настоящее время, несмотря на повышение компьютеризации общества, в сфере бизнеса и торговли до сих пор нет средств, позволяющих в достаточной мере автоматизировать процесс ведения документации и отчетности.

Целью данной дипломной работы является определение эффективности внедрения автоматизации документированной информации.

Также одной из основных задач можно рассматривать проблему ведения отчетности, а также оперативную корректировку данных при возникновении необходимости в этом.

О своевременности и актуальности рассматриваемой проблемы говорит тот факт, что большую часть своего времени администрация магазина тратит на оформление различной документации и отчетов. Огромное количество магазинов и отсутствие предложений в данной сфере гарантирует высокую потребность в данном продукте.

Также проблематику нынешнего мира можно рассмотреть и с другой стороны. Актуальность выбранной темы обусловлена тем, что компании осознали, что просто предлагать отличные продукты уже недостаточно: легкость копирования и насыщение рынка могут быстро свести на нет первые намеки на формулу успеха. Сегодня главным знаком отличия стала исключительность сервиса, предоставляемого на прочной и четкой основе. Сервис требует притока и вовлечения клиентов, и потому он значительно труднее, нежели продукт, поддается имитации. Следовательно, конкурентного преимущества можно достичь путем совместного применения знаний об ожиданиях, предпочтениях и поведении клиентов. А для этого необходим непрерывный диалог с клиентами и использование информации и идей, добываемых в каждой точке контакта с ними.  
Системы управления взаимоотношениями с клиентами, которую часто называют также менеджментом клиентов, является бизнес-подходом. Его цель — создание, развитие и укрепление отношений с тщательно выбранными клиентами, увеличение выгоды клиента, повышение корпоративной прибыли, а значит, и максимизация прибыли инвесторов.

MINISO — это розничный торговец товарами для жизни в японском стиле, предлагающий высококачественные товары для дома, косметику и продукты питания по доступным ценам. Основанная генеральным директором Е Гуофу, MINISO с 2013 года открыла более 4200 магазинов в более чем 80 странах и регионах, включая США, Великобританию, Канаду, Австралию, Испанию, ОАЭ, Индию и Мексику.

В MINISO вы можете приобрести:

1. Мелкие товары для дома (посуду, боксы, вешалки, полотенца и т.д).

2. Бытовую химию.

3. Косметику и косметические аксессуары.

4. Канцелярию (блокноты, ручки, фломастеры).

5. Детские игрушки (плюшевые зайцы и развивающие игры).

6. Аксессуары (сумки, кошельки, очки, часы).

7. Бижутерию (серьги, цепочки, резинки для волос, заколки).

8. Одежду (сланцы, головные уборы, нижнее белье, носки).

9. Массажеры, эспандеры.

10. Различную электронику (весы, будильники, аксессуары для телефона и компьютера, селфипалки).

11. Снеки и напитки (орехи, конфеты, соки).

MINISO — китайская компания, которую еще 10 лет назад считали не более, чем складом китайских подделок. За последние 5 лет она стала новым феноменом на рынке, открыв 2600 магазинов в 60 странах мира, включая даже Северную Корею. В июле 2013 года японский дизайнер Джуниа Мияке и китайский бизнесмен Е Гофу решили открыть компанию, которая доказала бы всем, что качественные товары на каждый день не обязательно должны быть дорогими.

Тема моей курсовой работы – база данных «Магазин «Miniso»». В магазине много товаров, которые предоставляются клиентам и с помощью этой базы будет систематизирована информация о товарах, клиентах и фирме, производящая товар. Данная база данных актуальна для любого магазина, занимающегося продажей товаров для дома, косметики и продуктов питания, так как с помощью этой базы данных покупатель может с легкостью найти информацию об интересующем его товаре и заказать его, а сотрудник магазина сможет без особых сложностей заказать распроданный товар у производителя. Данная база данных предлагает введение отчетности, хранения данных, ввод и корректировку данных. Базу данных могут использовать не только администрация интернет-магазина, но также и рабочий персонал магазина.

Таким образом, данная база данных является актуальной в сфере торговли товаров для жизни. Данная база данных выполнена в наиболее популярной системе управления базами данных – Access.

1.2 Описание языка программирования

1.2.1 Общие сведения

C# (произносится *си-шарп*, иногда переводят *си-диез*)  — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Java, Delphi, Модула и Smalltalk — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# не поддерживает множественное наследование классов (в отличие от C++).

Символ # в названии языка печатается на клавиатуре как Shift+З, что символизирует третью реализацию С. С другой стороны # можно интерпретировать и как две пары плюсов ++;++, намекающие на новый шаг в развитии языка по сравнению с C++ (подобно шагу от С к C++), и как музыкальный символ диез, вместе с буквой С составляющий в английском языке название ноты до-диез (англ. С sharp). # часто называют «шарпом» (от англ, sharp) из-за его схожести с диезом ?, отсюда и название языка —«Си шарп».

Актуальность: Язык программирования C# является одним из наиболее популярных за счет простоты его изучения.

История возникновения.

Язык C# появился на свет в июне 2000 г. в результате кропотливой работы большой группы разработчиков компании Microsoft, возглавляемой Андерсом Хейлсбергом (Anders Hejlsberg). Появление языка C# инициативы .NET отнюдь не случайно пришлось на начало лета 2000 г. Именно к этому моменту компания Microsoft подготовила промышленные версии новых компонентных технологий и решений в области обмена сообщениями и данными, а также создания Internet-приложений (COM+, ASP+, ADO+, SOAP, Biztalk Framework). Несомненно, лучшим способом продвижения этих новинок является создание инструментария для разработчиков с их полноценной поддержкой. В этом и заключается одна из главных задач нового языка C#. Кроме того, Microsoft не могла больше расширять все те же инструменты и языки разработки, делая их все более и более сложными (а это было необходимо, так как требования поддержки современного оборудования и требования обеспечения обратной совместимости с программными продуктами начала 90-х годов становились всё более конфликтующими). Наступил момент, когда необходимо начать с чистого листа для того, чтобы создать простой, но имеющий сложную структуру набор языков, сред и средств разработки, которые позволят разработчику легко создавать современные программные продукты. С# и .NET являются той самой отправной точкой. Если говорить упрощенно, то .NET представляет собой новую платформу, новый API (англ. Application Programming Interface - интерфейс прикладного программирования) для программирования в Windows, а С# - новый язык, созданный с нуля, для работы с этой платформой, а также для извлечения всех выгод из прогресса сред разработки и достижений объектно-ориентированного программирования в течение последних 20 лет.

Последнее время С и С++ являются наиболее используемыми языками для разработки коммерческих и бизнес приложений. Эти языки устраивают многих разработчиков, но в действительности не обеспечивают должной продуктивности разработки. К примеру, процесс написания приложения на С++ зачастую занимает значительно больше времени, чем разработка эквивалентного приложения, скажем, на Visual Basic. Сейчас существуют языки, увеличивающие продуктивность разработки за счет потери в гибкости, которая так привычна и необходима программистам на С/С++. Подобные решения являются весьма неудобными для разработчиков и зачастую предлагают значительно меньшие возможности. Эти языки также не ориентированы на взаимодействие с появляющимися сегодня системами и очень часто они не соответствуют существующей практике программирования для Web. Многие разработчики хотели бы использовать современный язык, который позволял бы писать, читать и сопровождать программы с простотой Visual Basic и в то же время давал мощь и гибкость C++, обеспечивал доступ ко всем функциональным возможностям системы, взаимодействовал бы с существующими программами и легко работал с возникающими Web стандартами.

Учитывая все подобные пожелания, Microsoft разработала новый язык - C#. В него входит много полезных особенностей - простота, объектная ориентированность, типовая защищенность, "сборка мусора", поддержка совместимости версий и многое другое. Данные возможности позволяют быстро и легко разрабатывать приложения, особенно COM+ приложения и Web сервисы. При создании C#, его авторы учитывали достижения многих других языков программирования: C++, C, Java, SmallTalk, Delphi, Visual Basic и т.д. Надо заметить что по причине того, что C# разрабатывался с чистого листа, у его авторов была возможность (которой они явно воспользовались), оставить в прошлом все неудобные и неприятные особенности (существующие, как правило, для обратной совместимости), любого из предшествующих ему языков. В результате получился действительно простой, удобный и современный язык, по мощности не уступающий С++, но существенно повышающий продуктивность разработок.

Особенности языка

C# разрабатывался как язык программирования прикладного уровня для CLR и, как таковой, зависит, прежде всего, от возможностей самой CLR. Это касается, прежде всего, системы типов C#, которая отражает BCL. Присутствие или отсутствие тех или иных выразительных особенностей языка диктуется тем, может ли конкретная языковая особенность быть транслирована в соответствующие конструкции CLR. Так, с развитием CLR от версии 1.1 к 2.0 значительно обогатился и сам C#; подобного взаимодействия следует ожидать и в дальнейшем. (Однако эта закономерность была нарушена с выходом C# 3.0, представляющим собой расширения языка, не опирающиеся на расширения платформы .NET.) CLR предоставляет C#, как и всем другим .NET-ориентированным языкам, многие возможности, которых лишены «классические» языки программирования. Например, сборка мусора не реализована в самом C#, а производится CLR для программ, написанных на C# точно так же, как это делается для программ на VB.NET, J# и др.

Реализация

Существует несколько реализаций C#:

* Реализация C# в виде компилятора csc.exe включена в состав .NET Framework.
* Реализация в Microsoft Silverlight.
* В составе проекта Rotor (Shared Source Common Language Infrastructure) компании Microsoft.
* Проект Mono включает в себя реализацию C# с открытым исходным кодом.
* Проект DotGNU также включает компилятор C# с открытым кодом.
* DotNetAnywhere[15] — ориентированная на Embedded-системы реализация CLR, поддерживает практически всю спецификацию C# 2.0.

Хотелось бы подчеркнуть современное удобство C#. Когда вы начнете работу с C#, а, надеюсь, это произойдет как можно скорее, вы увидите, что довольно большое значение в нем имеют пространства имен. Уже сейчас, на основе первого примера, вы можете судить об этом - ведь все файлы заголовков заменены именно пространством имен. Так в C#, помимо просто выражения using, предоставляется еще одна очень удобная возможность - использование дополнительного имени (alias) пространства имен или класса.

Современность C# проявляется и в новых шагах к облегчению процесса отладки программы. Традиционным средством для отладки программ на стадии разработки в C++ является маркировка обширных частей кода директивами #ifdef и т.д. В C#, используя атрибуты, ориентированные на условные слова, вы можете куда быстрее писать отлаживаемый код.

В наше время, когда усиливается связь между миром коммерции и миром разработки программного обеспечения, и корпорации тратят много усилий на планирование бизнеса, ощущается необходимость в соответствии абстрактных бизнес процессов их программным реализациям. К сожалению, большинство языков реально не имеют прямого пути для связи бизнес логики и кода. Например, сегодня многие программисты комментируют свои программы для объяснения того, какие классы реализуют какой-либо абстрактный бизнес объект. C# позволяет использовать типизированные, расширяемые метаданные, которые могут быть прикреплены к объекту. Архитектурой проекта могут определяться локальные атрибуты, которые будут связанны с любыми элементами языка - классами, интерфейсами и т.д. Разработчик может программно проверить атрибуты какого-либо элемента. Это существенно упрощает работу, к примеру, вместо того чтобы писать автоматизированный инструмент, который будет проверять каждый класс или интерфейс, на то, является ли он действительно частью абстрактного бизнес объекта, можно просто воспользоваться сообщениями основанными на определенных в объекте локальных атрибутах.

C#, являясь последним из широко распространенных языков программирования, должен впитать в себя весь имеющийся опыт и вобрать лучшие стороны существующих языков программирования, при этом являясь специально созданным для работы в .NET. Сама архитектура .NET продиктовала ему (как и многим другим языкам, на которых можно писать под .NET) объектно-ориентированную направленность. Конечно, это не является правилом, возможно создание компиляторов даже функциональных языков по .NET, на эту тему существуют специальные работы.

Свой синтаксис C# во многом унаследовал от C++ и Java. Разработчики, имеющие опыт написания приложений на этих языках, найдут в C# много знакомых черт. Но вместе с тем он является во многом новаторским - атрибуты, делегаты и события, прекрасно вписанные в общую идеологию языка, прочно заняли место в сердцах .NET - разработчиков. Их введение позволило применять принципиально новые приемы программирования.

Конечно, излюбленным объектом для сравнения с C# у мировой коммьюнити является Java. Также разработанный для работы в виртуальной среде выполнения, имеющей объектно-ориентированную архитектуру и сборщик мусора, основанный на механизме ссылок. При сравнении с этим языком сразу выделаются такие особенности, как возможность объявлять несколько классов в одном файле, из чего следует синтаксическая поддержка иерархической системы пространств имен. Из реализации ООП-концепций сходство в механизме наследования и реализации (и в Java и в C# возможно единичное наследование, но множественная реализация интерфейсов, в отличие от C++). Но в Java отсутствуют свойства и индексаторы (а также делегаты и события, но они отсутствуют еще много где). Также есть возможность перечисления контейнеров.

Из вещей, включенных в спецификацию языка, но не являющихся чисто "программистскими" необходимо отметить возможность использование комментариев в формате XML. Если комментарии отвечают специально описанной структуре, компилятор по ним может сгенерировать единый XML-файл документации.

Но C# внес и свои уникальные черты, которые уже были упомянуты - это события, индексаторы, атрибуты и делегаты. Все эти элементы будут обсуждены в следующих частях, сейчас лишь отмечу, что они предоставляют собой очень полезные возможности, которые не останутся невостребованными.

Ввиду очень удобного объектно-ориентированного дизайна, C# является хорошим выбором для быстрого конструирования различных компонентов - от высокоуровневой бизнес-логики до системных приложений, использующих низкоуровневый код. Также следует отметить, что C# является и Web ориентированным - используя простые встроенные конструкции языка ваши компоненты могут быть легко превращены в Web сервисы, к которым можно будет обращаться из Internet посредством любого языка на любой операционной системе. Дополнительные возможности и преимущества перед другими языками приносит в C# использование передовых Web технологий, таких как: XML (Extensible Markup Language) и SOAP (Simple Object Access Protocol). Среда разработки Web сервисов позволяет программисту смотреть на существующие сегодня Web приложения, как на родные C# объекты, что дает возможность разработчикам соотнести имеющиеся Web сервисы с их познаниями в объектно-ориентированном программировании.

База данных - набор сведений, хранящихся некоторым упорядоченным способом. Можно сравнить базу данных со шкафом, в котором хранятся документы. Иными словами, база данных — это хранилище данных. Сами по себе базы данных не представляли бы интереса, если бы не было систем управления базами данных (СУБД).

Данные — это формализованное представление информации, доступное для обработки, интерпретации и обмена между людьми или в автоматическом режиме.

Базы данных (БД) составляют в настоящее время основу компьютерного обеспечения информационных процессов, входящих практически во все сферы человеческой деятельности. Действительно, процессы обработки информации имеют общую природу и опираются на описание фрагментов реальности, выраженное в виде совокупности взаимосвязанных данных. Базы данных являются эффективным средством представления структур данных и манипулирования ими. Концепция баз данных предполагает использование интегрированных средств хранения информации, позволяющих обеспечить централизованное управление данными и обслуживание ими многих пользователей. При этом БД должна поддерживаться в среде ЭВМ единым программным обеспечением, называемым системой управления базами данных (СУБД).

Система управления базами данных — это совокупность языковых и программных средств, которая осуществляет доступ к данным, позволяет их создавать, менять и удалять, обеспечивает безопасность данных и т.д. В общем СУБД — это система, позволяющая создавать базы данных и манипулировать сведениями из них. А осуществляет этот доступ к данным СУБД посредством специального языка - SQL.

Одной из актуальных проблем любого вида человеческой деятельности является проблема обработки увеличивающегося потока информации. Перспективным способом решения данной проблемы является автоматизация работы с информацией, в частности, создание компьютерных баз данных, позволяющих хранить, систематизировать и обрабатывать данные.

Системы управления базами данных можно классифицировать по способу установления связей между данными, характеру выполняемых ими функций, сфере применения, числу поддерживаемых моделей данных, характеру используемого языка общения с базой данных и другим параметрам.

Классификация БД

Существует огромное количество разновидностей баз данных, отличающихся по различным критериям. Рассмотрим основные классификации:  
по выполняемым функциям СУБД подразделяются на операционные и информационные;  
по сфере применения СУБД подразделяются на универсальные и проблемно-ориентированные;  
по используемому языку общения СУБД подразделяются на замкнутые, имеющие собственные самостоятельные языки общения пользователей с базами данных, и открытые, в которых для общения с базой данных используется язык программирования, расширенный операторами языка манипулирования данными;  
по числу поддерживаемых уровней моделей данных СУБД подразделяются на одно-, двух-, трехуровневые системы; по способу установления связей между данными различают реляционные, иерархические и сетевые базы данных;  
по способу организации хранения данных и выполнения функций обработки базы данных подразделяются на централизованные и распределенные.

Компьютерная база данных — это хранилище объектов. В одной базе данных может быть больше одной таблицы. Например, система отслеживания складских запасов, в которой используются три таблицы, — это не три базы данных, а одна. В базе данных Access (если ее специально не настраивали для работы с данными или кодом, принадлежащими другому источнику) все таблицы хранятся в одном файле вместе с другими объектами, такими как формы, отчеты, макросы и модули.

Использование Access позволяет:

* добавлять новую информацию в базу данных, например новый артикул складских запасов;
* изменять информацию, уже находящуюся в базе, например перемещать артикул;
* удалять информацию, например если артикул был продан или утилизирован;
* упорядочивать и просматривать данные различными способами;
* обмениваться данными с другими людьми с помощью отчетов, сообщений электронной почты, внутренней сети или Интернета.

**Взаимосвязи данных**

Взаимосвязи данных могут быть одного из четырех типов:

1. один к одному;
2. один ко многим;
3. много к одному;
4. много ко многим

Давайте рассмотрим принципы построения таких взаимосвязей.

1. Вид взаимосвязи **один к одному** подразумевает, что каждая запись одного объекта базы данных будет указывать на единственную запись другого объекта. Например, с одним клиентом может быть связан только один заказ.
2. Взаимосвязь **один ко многим** означает, что одной записи объекта базы данных будет соответствовать несколько записей других объектов. Например, один клиент может иметь несколько квартир. Тогда с записью клиента будут связаны записи о его квартирах. Эта самая распространенная связь между базами данных. Предположим, нам нужно реализовать некую БД, которая ведет учет данных о пользователях. У пользователя есть: имя, фамилия, возраст, номера телефонов. При этом у каждого пользователя может быть от одного и больше номеров телефонов (многие номера телефонов). В этом случае мы наблюдаем следующее: пользователь может иметь многие номера телефонов, но нельзя сказать, что номеру телефона принадлежит определенный пользователь.  
   Другими словами, телефон принадлежит только одному пользователю. А пользователю могут принадлежать 1 и более телефонов (многие).
3. Вид взаимосвязи **много к одному** равносилен рассмотренному выше виду «один ко многим» и отличается от него только направлением. Представим, что нам нужно написать БД, которая будет хранить работником IT-компании. При этом существует некий стандартный набор должностей. При этом:

» Работник может иметь одну и более должностей. Например, некий работник может быть и админом, и программистом.

» Должностью может «владеть» одним и более работников. Например, админами является определенный набор работников. Другими словами, к админам относятся некие работники.

Работников представляет таблица «Employee» (id, имя, возраст), должности представляет таблица «Position» (id и название должности). Как видно, обе эти таблицы связаны между собой по правилу многие ко многим: каждому работнику соответствует одна и больше должностей (многие должности), каждой должности соответствует один и больше работников (многие работники).

Для реализации связи многие ко многим нам нужен некий посредник между двумя рассматриваемыми таблицами. Он должен хранить два внешних ключа, первый из которых ссылается на первую таблицу, а второй — на вторую.

1. Последний вид взаимосвязи **много ко многим** устанавливается между двумя типами объектов базы данных. Например, когда у одного банкира может быть несколько клиентов и, одновременно, один клиент может пользоваться услугами нескольких банков.

По принципу хранения данных все базы данных разделяются на несколько основных типов:

* иерархические;
* сетевые;
* реляционные.

## **Иерархические базы данных**

Иерархическая модель базы данных состоит из объектов с указателями от родительских объектов к потомкам, соединяя вместе связанную информацию. Иерархические БД могут быть представлены как дерево, состоящее из объектов различных уровней. Верхний уровень занимает один объект, второй - объекты второго уровня и т.д.;

Сетевая модель базы данных подобна иерархической, за исключением того, что в ней имеются указатели в обоих направлениях, которые соединяют родственную информацию;

Реляционная модель - «реляционный» от англ. Relation (отношение), ориентирована на организацию данных в виде двумерных таблиц, называемых еще реляционными таблицами. Информация, введенная в одну таблицу, может быть связана с одной или несколькими записями другой таблицы.

Реляционная модель данных (РМД) некоторой предметной области представляет собой набор отношений, изменяющихся во времени. При создании информационной системы совокупность отношений позволяет хранить данные об объектах предметной области и моделировать связи между ними. Термины РМД представлены в таблице 1.1.

Таблица 1.1 - Термины реляционной модели

|  |  |  |
| --- | --- | --- |
| Термин реляционной модели | Эквивалентный | термин |
| Отношение | Таблица | |
| Схема отношения | Строка заголовков столбцов таблицы (заголовок таблицы) | |
| Кортеж | Строка таблицы, запись | |
| Сущность | Описание свойств объекта | |
| Атрибут | Столбец, поле | |
| Домен | Множество допустимых значений атрибута | |
| Первичный ключ | Уникальный идентификатор | |
| Кардинальность | Количество строк | |
| Степень | Количество столбцов | |

Реляционная база данных представляет собой хранилище данных, содержащее набор двухмерных таблиц. Данные в таблицах должны удовлетворять следующим принципам:

1. Значения атрибутов должны быть атомарными (иными словами, каждое значение, содержащееся на пересечении строки и колонки, должно быть не расчленяемым на несколько значений).

2. Значения каждого атрибута должны принадлежать к одному и тому же типу.

3. Каждая запись в таблице уникальна.

4. Каждое поле имеет уникальное имя.

5. Последовательность полей и записей в таблице не существенна.

Отношение является важнейшим понятием и представляет собой двумерную таблицу, содержащую некоторые данные. Сущность есть объект любой природы, данные о котором хранятся в базе данных. Данные о сущности хранятся в отношении. Атрибуты представляют собой свойства, характеризующие сущ­ность. В структуре таблицы каждый атрибут именуется и ему соответ­ствует заголовок некоторого столбца таблицы.

Ключом отношения называется совокупность его атрибутов, однозначно идентифицирующих каждый из кортежей отношения. Иными словами, множество атрибутов К, являющееся ключом отношения, обладает свойством уникальности. Следующее свойство ключа — не избыточность. То есть никакое из собственных подмножеств множества К не обладает свойством уникальности. Каждое отношение всегда имеет комбинацию атрибутов, которая может служить ключом. Ее существование гарантируется принципом № 3 РМД. По крайней мере, вся совокупность атрибутов обладает свойством уникальности. Возможны случаи, когда отношение имеет несколько комбинаций атрибутов, каждая из которых однозначно определяет все кортежи отношения. Все эти комбинации атрибутов являются возможными ключами отношения. Любой из возможных ключей может быть выбран как первичный. Ключи обычно используют для достижения следующих целей:

» Исключения дублирования значений в ключевых атрибутах (осталь­ные атрибуты в расчет не принимаются);

» Упорядочения кортежей. Возможно упорядочение по возраста­нию или убыванию значений всех ключевых атрибутов, а также смешанное упорядочение (по одним — возрастание, а по другим — убывание);

» Организации связывания таблиц.

**Сетевые базы данных**

Сетевая база данных – это модель данных, где несколько записей или файлов могут быть связаны с несколькими владельцами файлов и наоборот. Модель может рассматриваться как перевернутое дерево, где каждый член – это отрасли, связанные с владельцем, который находится в нижней части дерева. По сути, это отношения в чистой форме, где один элемент может указывать на множество элементов данных, и само по себе может быть указано несколько элементов данных.

Модель сетевой базы данных позволяет каждой записи иметь несколько родителей и несколько дочерних записей, которые, когда они визуализируются, принимают форму сетевой структуры сетевых записей. В отличие от [иерархической модели данных](https://mega-obzor.ru/ierarxicheskaya-baza-dannyx.html) она может иметь только одну родительскую запись, но может иметь много дочерних записей.

От всех этих сложностей избавлены клиент-серверные базы данных. Клиент-серверные базы данных (удаленные базы данных) — программы СУБД располагаются в оперативной памяти рабочих станций локальной сети, а файлы базы данных - на магнитных дисках файл-сервера. Специальный интерфейсный модуль распознает, где находятся файлы, к которым осуществляется обращение. В связи с этим данная СУБД может работать как с локальными базами данных, так и с центральной базой данных. Синхронизация совместного использования базы данных файл-сервера возлагается на систему управления базами данных, которая должна обеспечивать блокирование записей на время их корректировки, чтобы сделать их недоступными с других рабочих станций.

Использование файл-серверов предполагает, что вся обработка данных выполняется на рабочей станции, а файл-сервер лишь выполняет функции накопителя данных и средств доступа.

Клиент-серверные базы данных работают с помощью запросов на языке SQL (Structured Query Language).

Для работы с различными реляционными базами данных, включая Oracle, MySQL, PostgreSQL, DBase, FoxPro, Clipper, Paradox был создан единый язык запросов к базам данных. Назвали его язык SQL, что означает Structured Query Language — структурированный язык запросов.

SQL-запросы применяются для любых операций с таблицами базы данных, расположенной на сервере.

При помощи запросов SQL можно:

* Создавать таблицы БД;
* Изменять таблицы БД;
* Удалять таблицы БД;
* Вставлять записи (строки) в таблицы БД;
* Редактировать записи в таблицах БД;
* Извлекать выборочную информацию из таблиц БД;
* Удалять выборочную информацию из БД.

По видам запросы SQL чаще всего делятся на:

* запросы, предназначенные для работы со структурой данных - для создания, описания и модификации БД;
* запросы, используемые непосредственно в работе с данными, с помощью которых можно добавлять, обновлять, сохранять и удалять данные;
* запросы, применяемые для предоставления или отмены прав доступа к БД;

В свою очередь, каждый из видов SQL-запросов подразделяется на типы:

* команды, работающие со структурой БД. К ним относятся CREATE -  «создать» (например, CREATE TABLE (создать таблицу), CREATE USER (создать пользователя)), ALTER - «модифицировать» (этот запрос используется при внесении изменений в саму БД или в ее часть), DROP - «удалить» (также относятся к БД и ее частям);
* команды, работающие с данными. К наиболее востребованным запросам относятся: SELECT (выборка данных), INSERT (вставка новых данных), UPDATE (обновление данных), DELETE (удаление данных), MERGE (слияние данных);
* команды, работающие с правами доступа. В их список входят GRANT - разрешение пользователю на проведение определенных операций с БД или данными; REVOKE – отзыв выданного  разрешения; DENY – установка запрета, имеющего приоритет над разрешением.

При составлении SQL-запроса для работы с базами данных в СУБД (MySQL, Microsoft SQL Server, PostgreSQL) вводятся следующие параметры отбора:

* названия таблиц, из которых необходимо извлечь данные;
* поля, значения которых требуется вернуть к исходным после внесения изменений в БД;
* связи между таблицами;
* условия выборки;
* вспомогательные критерии отбора (ограничения, способы представления информации, тип сортировки).

В клиент-серверных приложениях баз данных вся информационная система делится на две разнородные части:

1. клиент базы данных;
2. сервер базы данных.

Клиент базы данных — это приложение пользователя, которое обеспечивает формирование и передачу запросов серверу, а также отображение полученных результатов. Кроме того, приложение-клиент должно обеспечивать интерфейс пользователя базы данных.

Сервер базы данных — специальная программа, которая управляет удаленной базой данных и обеспечивает выполнение запросов клиентов и выдачу результатов запросов.

Приложения, работающие с клиент-серверными базами данных, называются двухъярусными или многоярусными.

Как видно из названия, главные «действующие лица»:

клиент – компьютерное устройство, которое отсылает запросы серверу, касающиеся выполнения определенных задач или предоставления конкретной информации.

сервер – компьютерное устройство, гораздо мощнее обычного ПК.

Система работает по следующему принципу:

1. Клиент отправляет запрос серверной машине.

2. Сервер принимает обращение с требованием выполнить определенное действие и выполняет поставленную задачу.

3. Программно-аппаратный комплекс отправляет клиенту результат выполненной работы, обработанного запроса.

Модель клиент-сервер предоставляет возможность разграничить поставленные задачи и работу над вычислениями между теми, кто заказывает услуги и теми, кто их поставляет.

Основные компоненты системы:

Клиент - рабочая станция считается входной точкой конечного пользователя в данной системе. Отправляет запросы, получает ответы;

Сервер - взаимодействует с многочисленными клиентами и решает поставленные ими задачи;

Сеть - здесь происходит передача данных. Посредством сети можно соединить рабочие машины общими ресурсами;

Приложения - могут обрабатывать информацию, организовывать физическое распределение данных между сервером и клиентом. Программным обеспечением оснащают серверные устройства для сбора данных, работы с ними и хранения. А также ПО устанавливают на компьютерной станции-клиенте.

Двухуровневая клиент-серверная архитектура, использующая технологию «клиент-сервер». Клиент-серверная СУБД позволяет обмениваться клиенту и серверу минимально необходимыми объёмами информации. При этом основная вычислительная нагрузка ложится на сервер.

DB-сервер отличается от файл-сервера тем, что в его оперативной памяти, помимо сетевой операционной системы, функционирует централизованная СУБД, которая обеспечивает совместное использование рабочими станциями базы данных, размещенной во внешней памяти этого DB-сервера.

Основной принцип технологии «Клиент-сервер» заключается в разделении функций приложения как минимум на три группы:

- модули интерфейса с пользователем. Также эту группу называют логикой представления. Через эту группу пользователи взаимодействуют с приложением. Независимо от конкретных характеристик логики представления (интерфейс командной строки, сложные графические пользовательские интерфейсы, интерфейсы через посредника) ее задача состоит в том, чтобы обеспечить средства для наиболее эффективного обмена информацией между пользователем и информационной системой.

- модули хранения данных. Эту группу также называют бизнес-логикой. Бизнес-логика определяет, для чего конкретно предназначено приложение (например, прикладные функции, характерные для данной предметной области). Разделение приложения по границам между программами обеспечивает естественную основу для распределения приложения на нескольких компьютерах.

- модули обработки данных (функции управления ресурсами). Эту группу также называют логикой доступа к данным или алгоритмами доступа к данным. Алгоритмы доступа к данным исторически рассматривались как специфический для конкретного приложения интерфейс к механизму постоянного хранения данных наподобие файловой системы или СУБД. При помощи модулей обработки данных организуется специфический для приложения интерфейс к СУБД. При помощи интерфейса приложение управляет соединениями с базой данных и запросами к ней (перевод специфических для конкретного приложения запросов на язык SQL, получение результатов и перевод этих результатов обратно в специфические для конкретного приложения структуры данных).

Каждая из этих групп может быть реализована независимо от двух других. Например, не изменяя программ, используемых для хранения и обработки данных, можно изменить интерфейс с пользователем таким образом, что одни и те же данные будут отображаться в виде таблиц, графиков или гистограмм. Очень простые приложения часто способны собрать все три части в единственную программу, и подобное разделение соответствует функциональным границам. В соответствии с разделением функций в любом приложении выделяются следующие компоненты:

- компонент представления данных;

- прикладной компонент;

- компонент управления ресурсом

Microsoft Access – мощное приложение, предлагающее широкий диапазон средств для эффективного управления информацией. Возможности Access постоянно расширяются. Программа ACCESS входит в пакет самого распространённого пакета офисных программ Microsoft. Эта программа универсальна в своём роде. При веем при этом, программа проста в эксплуатации и доступна рядовому пользователю. И за всё это отвечает интерфейс. В процессе совершенствования этой программы были внесены уникальные возможности. Данные могут быть представлены в виде таблиц или диаграмм. И если принять во внимание, что данной программой может пользоваться любой пользователь (от новичка до профессионального разработчика), то без сомнения можно утверждать что ACCESS корпорации Microsoft это лучшее средства для решения задач любой сложности. Система управления базами данных Microsoft Access является одним из самых популярных приложений в семействе настольных СУБД. Все версии Access имеют в своем арсенале средства, значительно упрощающие ввод и обработку данных, поиск данных и предоставление информации в виде таблиц, графиков и отчетов. Начиная с версии Access 2000, появились также Web-страницы доступа к данным, которые пользователь может просматривать с помощью программы Internet Explorer. Помимо этого, Access позволяет использовать электронные таблицы и таблицы из других настольных и серверных баз данных для хранения информации, необходимой приложению. Присоединив внешние таблицы, пользователь Access будет работать с базами данных в этих таблицах так, как если бы это были таблицы Access. При этом и другие пользователи могут продолжать работать с этими данными в той среде, в которой они были созданы.

Для выполнения почти всех основных операций Access предлагает большое количество Мастеров (Wizards), которые делают основную работу за пользователя при работе с данными и разработке приложений, помогают избежать рутинных действий и облегчают работу неискушенному в программировании пользователю.

Особенности MS Access, отличающиеся от представления об «идеальной» реляционной СУБД.

Создание многопользовательской БД Access и получение одновременного доступа нескольких пользователей к общей базе данных возможно в локальной одноранговой сети или в сети с файловым сервером. Сеть обеспечивает аппаратную и программную поддержку обмена данными между компьютерами. Access следит за разграничением доступа разных пользователей к БД и обеспечивает защиту данных. При одновременной работе. Так как Access не является клиент серверной СУБД, возможности его по обеспечению многопользовательской работы несколько ограничены. Обычно для доступа к данным по сети с нескольких рабочих станций, файл БД Access (с расширением \*.mdb) выкладывается на файловый сервер. При этом обработка данных ведется в основном на клиенте – там, где запущено приложение, в силу принципов организации файловых СУБД. Этот фактор ограничивает использование Access для обеспечения работы множества пользователей (более 15–20) и при большом количестве данных в таблицах, так как многократно возрастает нагрузка не сеть.

В плане поддержки целостности данных Access отвечает только моделям БД небольшой и средней сложности. В нем отсутствуют такие средства как триггеры и хранимые процедуры, что заставляет разработчиков возлагать поддержание бизнес логики БД на клиентскую программу.

В первую очередь можно отметить распространенность, которая обусловлена тем, что Access является продуктом компании Microsoft, программное обеспечение и операционные системы которой использует большая часть пользователей персональных компьютеров. MS Access полностью совместим с операционной системой Windows, постоянно обновляется производителем, поддерживает множество языков.

В целом MS Access предоставляет большое количество возможностей за сравнительно небольшую стоимость. Также необходимо отметить ориентированность на пользователя с разной профессиональной подготовкой, что выражается в наличии большого количества вспомогательных средств (Мастеров, как уже отмечалось), развитую систему справки и понятный интерфейс. Эти средства облегчают проектирование, создание БД и выборку данных из нее.

MS Access предоставляет в распоряжение непрограммирующему пользователю разнообразные диалоговые средства, которые позволяют ему создавать приложения не прибегая к разработке запросов на языке SQL или к программированию макросов или модулей на языке VBA.

Еще одно немаловажное преимущество MS Access заключается в развитых встроенных средствах разработки приложений. Большинство приложений, распространяемых среди пользователей, содержит тот или иной объем кода VBA (Visual Basic for Applications). Поскольку VBA является единственным средством для выполнения многих стандартных задач в Access (работа с переменными, построение команд SQL во время работы программы, обработка ошибок, использование Windows API ит. д.), для создания более-менее сложных приложений необходимо его знание и знание объектной модели MS Access.

Одним из средств программирования в Access является язык макрокоманд. Программы, созданные на этом языке, называются макросами и позволяют легко связывать отдельные действия, реализуемые с помощью форм, запросов, отчетов. Макросы управляются событиями, которые вызываются действиями пользователями при диалоговой работе с данными через формы или системными событиями.

Получается что Access, обладая всеми чертами СУБД, предоставляет и дополнительные возможности. Это не только гибкая и простая в использовании СУБД, но и система для разработки работающих с базами данных приложений.

В отличие от других настольных СУБД, Access хранит все данные в одном файле, хотя и распределяет их по разным таблицам, как и положено реляционной СУБД. К этим данным относится не только информация в таблицах, но и другие объекты базы данных, которые будут описаны ниже.

Основу базы данных составляют хранящиеся в ней данные. Кроме того, в базе данных Access есть другие важные компоненты, которые называются объектами.

Объектами Access являются:

• Таблицы – основной объект Access. Список таблиц, составляющих базу данных приложения, появляется в окне базы данных при первом открытии приложения. Кроме этого, Access создает системные таблицы, в которых хранится информация обо всех объектах приложения, и эти таблицы при необходимости также могут отображаться в окне базы данных.

• Запросы – позволяют задавать условия для отбора данных и вносить изменения в данные.

• Формы – позволяют просматривать и редактировать информацию.

• Страницы – файлы в формате HTML (Hypertext Markup Language, Язык разметки гипертекста), позволяющие просматривать данные Access с помощью браузера Internet Explorer.

• Отчеты – позволяют обобщать и распечатывать информацию.

• Макросы – выполняют одну или несколько операций автоматически.

Базой данных Access является файл, который имеет расширение mdb. Этот файл может содержать не только все таблицы, но и другие объекты приложений Access — запросы, формы, отчеты, страницы доступа к данным, макросы и модули.

Access может работать одновременно только с одной базой данных. Но одна БД Access может включать множество таблиц, форм, запросов, отчётов, макросов и модулей, которые хранятся в одном файле с расширением mdb. Access позволяет создавать структуру таблицы в трех режимах – в режиме конструктора, с помощью мастера и путем ввода данных. Разница предполагает использование этих средств пользователями с разным уровнем подготовки, разными целями и перспективами использования данных. Естественно имеется возможность просматривать, редактировать, удалять и добавлять записи, осуществлять поиск, замену, сортировку данных, изменять вид таблицы.

Связи между таблицами определяются специальным средством, которое называется «Схема данных». Это удобный графический инструмент, позволяющий создавать связи между определенными полями таблиц, задавать различные типы отношений, устанавливать ограничения ссылочной целостности. При этом изменения сразу применяются в базе данных (естественно, если содержащиеся данные удовлетворяют всем условиям) Полученную диаграмму таблиц и связей можно распечатать, что, несомненно, удобно для разработчика.

Одной из основных задач создания и использования баз данных является предоставление пользователям необходимой информации на основе существующих данных. В Access для этих целей предназначены формы и отчеты.

Форма – это специальный объект-контейнер для других интерфейсных компонентов, таких как поля ввода и отображения данных, кнопки и др. На форме разработчик располагает компоненты для ввода, корректировки, просмотра и группировки данных, в зависимости от специфики приложения. Форму также можно создать двумя способами – в режиме конструктора и с помощью мастера. В первом случае разработчик располагает набором компонентов (при необходимости можно зарегистрировать свои ActiveX), которые свободно размещает на форме и задает их параметры. В режиме мастера пользователь просто выбирает таблицу, поля и стиль оформления, а форма генерируется автоматически.

Отчёт – объект, предназначенный для создания документа, который впоследствии может быть распечатан либо включён в документ другого приложения. Этот документ содержит результаты выборки из базы данных в виде структурированной информации (например в виде таблицы или списка). MS Access обладает богатыми возможностями по оформлению и форматированию отчетов. Те же два режима создания есть и у отчета. В режиме мастера у пользователя имеется возможность задать множество параметров, практически полностью определяющих желаемый внешний вид отчета. Для создания нестандартных отчетов лучше пользоваться конструктором.

Страницы – средство публикации данных в локальной сети или Internet. Создаваемая страница проектируется подобно форме (с некоторыми отличиями в используемых компонентах), при работе с ней можно не только просматривать, но и изменять данные в базе. После сохранения страницы как объекта в БД ее можно экспортировать в виде файла в формате HTML и использовать для доступа к данным через интернет-браузер.

Макрос – это объект, представляющий собой последовательность макрокоманд для автоматизации наиболее часто выполняемых действий при работе с базой. Макрокоманды выбираются из имеющего списка, а параметры задаются разработчиком. Выполнить макрос можно по нажатию на кнопку и программным методом в коде.

Модуль – контейнер программного кода на VBA. Для их редактирования и просмотра используется оболочка Редактора Visual Basic. Весь программный код приложения содержится в наборе модулей. Здесь он имеет то же смысловое значение, как и в любом языке программирования[3].

Это полный список объектов, которые можно хранить в базе данных MS Access, который, впрочем, похож и в других настольных СУБД подобного класса, например Paradox.

Какие возможности предоставляет Access в функциональном плане.

Ввод данных. Ввод данных может осуществляться следующими способами: вручную прямо в таблицу (сюда же относится вставка содержимого буфера обмена); вручную в поля формы; прямой импорт данных из других источников (базы Access, текстовые файлы, формат DBF, электронные таблицы, источники данных ODBC); программным методом, который может сочетать в себе любые средства, которые возможно реализовать на VBA.

Последний способ обладает наибольшей гибкостью и представляет практически неограниченные возможности, однако он самый сложный в реализации и требует определенного уровня знаний программирования.

2. Изменение данных. Редактирование возможно следующими способами: вручную прямо в таблице; в полях форм; в окне браузера, в котором загружена web-страница из БД; программным методом.

3. Вывод данных. Здесь Access предоставляет такие возможности: вывод на экран монитора в табличном виде, полях форм или отчетов; экспорт в другие форматы данных (те же, что при импорте); вывод на печать, в основном в виде отчетов; вывод данных в интернет-браузер с помощью объекта Страница в пределах определенной сети; программный экспорт и вывод информации.

4. Взаимодействие с другими источниками и потребителями информации. В этом плане Access может выступать как сервер или клиент автоматизации. Особенно прозрачно настраиваются связи с другими продуктами пакета Microsoft Office. Например вы можете иметь документ Microsoft Word, в котором будут присутствовать поля из БД Access, изменение данных в базе автоматически отображается в документе. Это позволяет создавать гибкие решения, интегрирующие данные в офисных средствах. 5. Использование базы данных MS Access другими приложениями. Такой вариант использует файл MDB как хранилище данных. Программа, которая обращается к данным, может быть написана на любом языке высокого уровня. В данном случае используются таблицы и запросы. О поддержании ссылочной целостности и актуальности данных следит ядро БД. Взаимодействие происходит через ODBC-драйвер Microsoft Jet Engine.

6. Средства создания приложений. Создание приложений на Access во многом подобно всем остальным средствам автоматизации Microsoft Office. Здесь используется интерпретируемый язык Visual Basic for Applications, что приводит, как и при использовании любого интерпретируемого языка, к определенному увеличению затрат процессорного времени и уменьшению скорости работы программ и обработки данных. Для успешной разработки необходимо знать объектную модель самого Access и особенности ее использования.

7. Особенности управления данными. При работе с Базами Данных в многопользовательском режиме возникают ситуации, когда необходимо ограничить число обращающихся пользователей к данным. Это делается для того, чтобы предотвратить одновременное обновление одной и той же записи, при глобальном обновлении данных или при техническом обслуживания самой Базы Данных.