Министерство образования Республики Беларусь

Учреждение образования «Институт информационных технологий Белорусского государственного университета информатики и радиоэлектроники»

Факультет компьютерных технологий

Кафедра информационных систем и технологий

Дисциплина: Базы данных

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе

на тему

“Центр детского творчества”

БГУИР КП 1-40 01 01 10  ПЗ

Студент: гр. 68107042 Иванов К. Е.

Минск 2018

**Содержание**

[1. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ 4](#_Toc516568500)

[2. ПРОЕКТНАЯ ЧАСТЬ 8](#_Toc516568501)

[2.1 Анализ предметной области 8](#_Toc516568502)

[2.2 Инфологическое проектирование 11](#_Toc516568503)

[2.3. Обоснование выбора программных средств 15](#_Toc516568504)

[2.4. Даталогическое проектирование 18](#_Toc516568505)

[2.5. Физическое проектирование 21](#_Toc516568506)

[ЗАКЛЮЧЕНИЕ 26](#_Toc516568507)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 27](#_Toc516568508)

[ПРИЛОЖЕНИЕ А 28](#_Toc516568509)

**ВВЕДЕНИЕ**

Данная курсовая работа выполнена в соответствии с заданием на курсовое проектирование. Темой работы является «Центр детского творчества».

В современном мире стремительными темпами развиваются информационные технологии и те сферы человеческой деятельности, которые с ними связаны. С каждым годом всё больший и больший объём информации обрабатывается при помощи современных и продолжающих совершенствоваться персональных компьютерах.

Любая организация нуждается в своевременном доступе к информации. Ценность информации в современном мире очень высока. Роль распорядителей информации в современном мире чаще всего выполняют базы данных. Базы данных обеспечивают надежное хранение информации, структурированном виде и своевременный доступ к ней. Практически любая современная организация нуждается в базе данных, удовлетворяющей те или иные потребности по хранению, управлению и администрированию данных.

Главная цель, которая стоит перед написанием курсового проекта – это приобретение практических навыков проектирования базы данных «Центр детского творчества» с использованием программ СУБД. В ходе выполнения данного курсового проекта необходимо будет разработать информационную базу данных и программное обеспечение для работы совместно с базой данных для центра детского творчества, которые помогут пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени.

## 1. ТЕОРЕТИЧЕСКИЙ РАЗДЕЛ

База данных (БД) – это совокупность специальным образом организованных данных хранимых в памяти вычислительной системы отображающих состояние объектов и их взаимосвязи в рассматриваемой предметной области.

Предметной областью называется часть реальной системы, представляющая интерес для определенного исследования. Различать полную предметную область (предприятие) и организационную единицу этой предметной области (отделы).

Основные термины.

Объект – термин, обозначающий факт, лицо, событие, предмет, о котором могут быть собраны данные (каждая таблица представляет один объект);

Реляционная БД - основной тип современных баз данных. Состоит из таблиц, между которыми могут существовать связи по ключевым значениям.

Таблица базы данных (table) – регулярная структура, которая состоит из однотипных строк (записей, records), разбитых на столбцы (поля, fields).

В теории реляционных баз данных синоним таблицы – отношение (relation), в котором строка называется кортежем, а столбец называется атрибутом.

В концептуальной модели реляционной БД аналогом таблицы является сущность (entity), с определенным набором свойств - атрибутов, способных принимать определенные значения (набор допустимых значений - домен).

Ключевой элемент таблицы (ключ, regular key) - такое ее поле (простой ключ) или строковое выражение, образованное из значений нескольких полей (составной ключ), по которому можно определить значения других полей для одной или нескольких записей таблицы. На практике для использования ключей создаются индексы - служебная информация, содержащая упорядоченные сведения о ключевых значениях. В реляционной теории и концептуальной модели понятие "ключ" применяется для атрибутов отношения или сущности.

Первичный ключ (primary key) - главный ключевой элемент, однозначно идентифицирующий строку в таблице. Могут также существовать альтернативный (candidate key) и уникальный (unique key) ключи, служащие также для идентификации строк в таблице.

В реляционной теории первичный ключ - минимальный набор атрибутов, однозначно идентифицирующий кортеж в отношении.

В концептуальной модели первичный ключ - минимальный набор атрибутов сущности, однозначно идентифицирующий экземпляр сущности.

Связь (relation) - функциональная зависимость между объектами. В реляционных базах данных между таблицами устанавливаются связи по ключам, один из которых в главной (parent, родительской) таблице - первичный, второй - внешний ключ - во внешней (child, дочерней) таблице, как правило, первичным не является и образует связь "один ко многим" (1:N). В случае первичного внешнего ключа связь между таблицами имеет тип "один к одному" (1:1). Информация о связях сохраняется в базе данных.

Внешний ключ (foreign key) - ключевой элемент подчиненной (внешней , дочерней)

Для работы с БД используются системы управления базами данных (СУБД).

СУБД – комплекс языковых и программных средств, предназначенных для создания, ведения и совместного использования БД несколькими пользователями. СУБД позволяет: создавать БД; вставлять, обновлять, удалять и извлекать информацию из БД; предоставляет контролируемый доступ к базе данных.

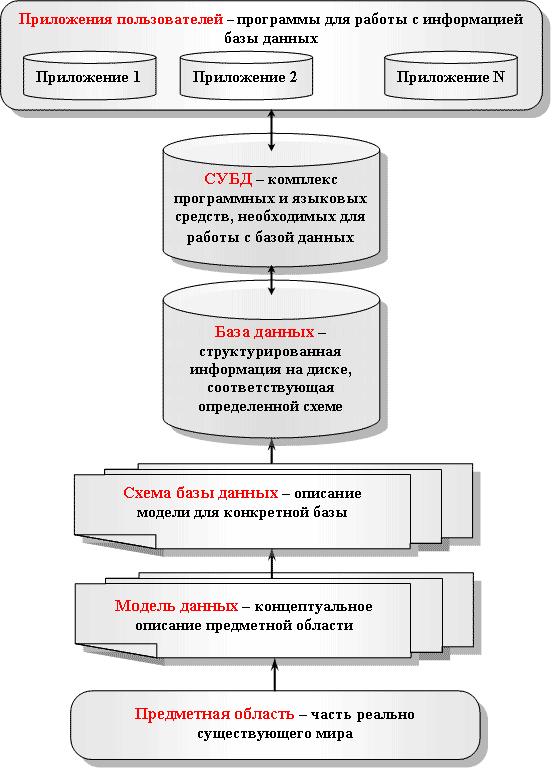


Рисунок 1.1. Взаимосвязь основных терминов в области проектирования баз данных и работы с ними

Процесс проектирования БД представляет собой процесс переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. Конечной целью проектирования является построение конкретной БД. Очевидно, что процесс проектирования сложен и поэтому имеет смысл разделить его на логически завершенные части – этапы.

Можно выделить пять основных этапов проектирования БД:

1. Сбор сведений и системный анализ предметной области - это первый и важнейший этап при проектировании БД. В нем необходимо провести подробное словесное описание объектов предметной области и реальных связей, присутствующих между реальными объектами.

2. Инфологическое проектирование - частично формализованное описание объектов предметной области в терминах некоторой семантической модели.

3. Выбор СУБД осуществляется на основе различных требований к БД и, соответственно, возможностей СУБД, а также в зависимости от имеющегося опыта разработчиков.

4. Даталогическое проектирование – это описание БД в терминах принятой даталогической модели данных. В реляционных БД даталогическое или логическое проектирование приводит к разработке схемы БД, т.е. совокупности схем отношений, которые адекватно моделируют объекты предметной области и семантические связи между объектами. Основой анализа корректности схемы являются функциональные зависимости между атрибутами БД.

5. Физическое проектирование заключается в увязке логической структуры БД и физической среды хранения с целью наиболее эффективного размещения данных, т.е. отображение логической структуры БД в структуру хранения.

## 2. ПРОЕКТНАЯ ЧАСТЬ

## 2.1 Анализ предметной области

Предметом курсового проекта является проектирование информационной базы данных и создание программного обеспечения для работы совместно с базой данных для центра детского творчества.

Программный продукт «Центр детского творчества» поможет пользователю легко найти нужную информацию о сотруднике, детях, группах и событиях в любом момент времени, автоматизировать все рутинные процессы по учету и управлению данными.

Сфера применения программы «Центр детского творчества» это администрирование детских развивающих центров, детских творческих клубов, центров раннего развития и предприятий подобной направленности.

С помощью программы «Центр детского творчества» можно будет выполнять следующие функции:

* Вести клиентскую базу детей с учетом их сопровождающих (родители, родственники)
* Вести клиентскую базу педагогов
* Проводить мероприятия
* В удобном виде регистрировать посещения детей на занятия
* Планировать расписание занятий на любой период
* Составлять график работы педагогов
* И много другое…

На сегодняшний день существует большое количество аналогов. Для сравнения функционального и визуального обеспечения программы «Центр детского творчества», были выбраны программы «Хеликс: Детский Центр» и «CRM-система для автоматизации детских центров Отмечалка».

«Хеликс: Детский Центр» - компьютерная программа, разработанная специально для детских развивающих центров, которая поможет автоматизировать все рутинные процессы по учету и управлению бизнесом.

"Хеликс: Детский Центр" разработан на платформе "1С:Предприятие 8.3". Из плюсов программы можно выделить то что она максимально простая и удобная в работе, а также в ней реализованы множество функциональных возможностей. Из минусов можно отметить, что «Хеликс: Детский Центр» является платной программой, для её установки требуется выезд специалиста, и гарантия на обслуживание составляет всего 3 месяца.

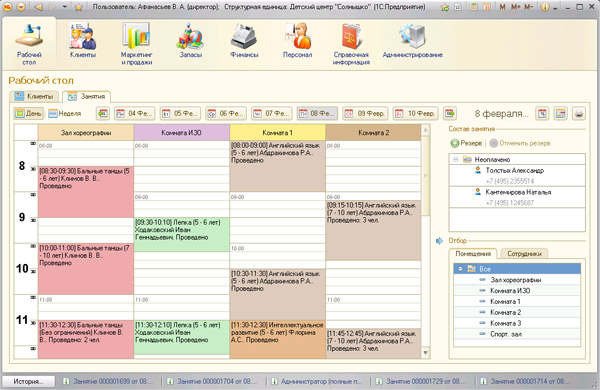


Рисунок 2.1.1 – Хеликс: Детский Центр»

«CRM-система для автоматизации детских центров Отмечалка» - это система автоматизации, предоставляющая полный набор инструментов, которые помогут раскрыть потенциал детского центра, повысить его прибыльность и конкурентоспособность.

Из плюсов это программы можно отметить то что в ней есть бесплатный 14 дневный период, постоянное обновление программы и высокая надёжность хранения данных.

В минусы этой программы также входят можно записать необходимость платить за её использование, а также необходимость подключения к сети интернет.

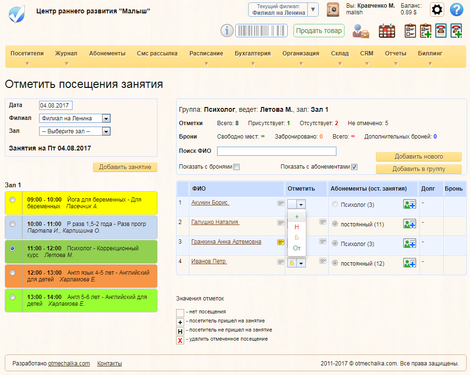


Рисунок 2.1.2 – «CRM-система Отмечалка»

Сравнительный анализ показал, что «Центр детского творчества», разрабатываемый в рамках курсового проектирования, по основным характеристикам не уступает существующим аналогам.

Входной информацией в создаваемой программе будут являться запросы пользователей.

Выходной информацией является подробная информация о детях, сотрудниках, группах и событиях представленная в виде таблиц.

Для достижения цели курсового проекта необходимо решить следующие задачи:

* анализ предметной области;
* построение концептуальной модели базы данных;
* организация базы данных;
* разработка прикладной программы;
* наполнение и сопровождение базы данных;

## 2.2 Инфологическое проектирование

Цель инфологического этапа проектирования состоит в получении семантических (концептуальных) моделей, отражающих предметную область и информационные потребности пользователей. В качестве инструмента для построения семантических моделей данных на этапе инфологического проектирования является неформальная модель "Сущность-Связь" (ER-модель - Entity-Relationship). Моделирование предметной области базируется на использовании графических диаграмм, включающих небольшое число разнородных компонентов.

Основными понятиями ER-модели являются сущность, связь и атрибут.

Сущность (объект) - это реальный или представляемый объект предметной области, информация о котором должна сохраняться и быть доступна. Различают такие понятия, как тип сущности и экземпляр сущности. Понятие тип сущности относится к набору однородных предметов, событий, личностей, выступающих как единое целое. Экземпляр сущности относится к конкретной вещи в наборе. В диаграммах ER-модели сущность представляется в виде прямоугольника (в нотации Баркера), содержащего имя сущности.

Выделим базовые сущности данного курсового проекта:

* **Дети**. Атрибуты: информация о детях, родители...
* **События**. Атрибуты – описание, дата проведения, заметки, цена.
* **Группы**. Атрибуты – сотрудник, класс, расписание, размер группы, цена.
* **Сотрудники**. Атрибуты – ФИО, дата рождения, квалификация.

**Родители** и **Информацию о детях** будем рассматривать как связь с детьми. Атрибуты родителей: ФИО мамы, ФИО папы, дата рождения мамы, дата рождения папы. Атрибуты информации о детях: ФИО, дата рождения, адрес.

**Класс** будем рассматривать как связь с группой. Атрибуты класса – название, описание, заметки, категория. А **Категории** рассматриваем как связь с классом. Атрибуты категории – название, описание.

**Лист** в данной работе рассматривается как связь между такими сущностями как дети, события и группы.

Атрибут - поименованная характеристика сущности, определяющая его свойства и принимающая значения из некоторого множества значений. Каждый атрибут обеспечивается именем, уникальным в пределах сущности.

Атрибуты могут классифицироваться по принадлежности к одному из трех различных типов: описательные, указывающие, вспомогательные. Описательные атрибуты представляют факты, внутренне присущие каждому экземпляру сущности. Указывающие атрибуты используются для присвоения имени или обозначения экземплярам сущности. Вспомогательные атрибуты используются для связи экземпляра одной сущности с экземпляром другого. Атрибуты подчиняются строго определенным правилам.

Множество из одного или нескольких атрибутов, значения которых однозначно определяют каждый экземпляр сущности, называются идентификатором. Каждый экземпляр сущности должен иметь хотя бы один идентификатор. Если идентификаторов несколько, один из них выбирается как привилегированный.

Связь (Relationship) - это поименованная графически изображаемая ассоциация, устанавливаемая между сущностями и представляющая собой абстракцию набора отношений, которые систематически возникают между различными видами предметов в реальном мире. Большинство связей относятся к категории бинарных и имеют место между двумя сущностями.

Среди бинарных связей существуют три фундаментальных вида связи: один-к-одному (1:1), один-ко-многим (1:M), многие-ко-многим (M:M). Связь один-к-одному (1:1) существует, когда один экземпляр одной сущности связан с единственным экземпляром другой сущности. Связь один-ко-многим (1:M) имеет место, когда один экземпляр одной сущности связан с одним или более экземпляром другой сущности и каждый экземпляр второй сущности связан только с одним экземпляром первой сущности. Связь многие-ко-многим (М:N) существует, когда один экземпляр одной сущности связан с одним или более экземпляром другой сущности и каждый экземпляр второй сущности связан с одним или более экземпляром первой сущности.

В данной курсовой работе предоставлены все три вида связей.

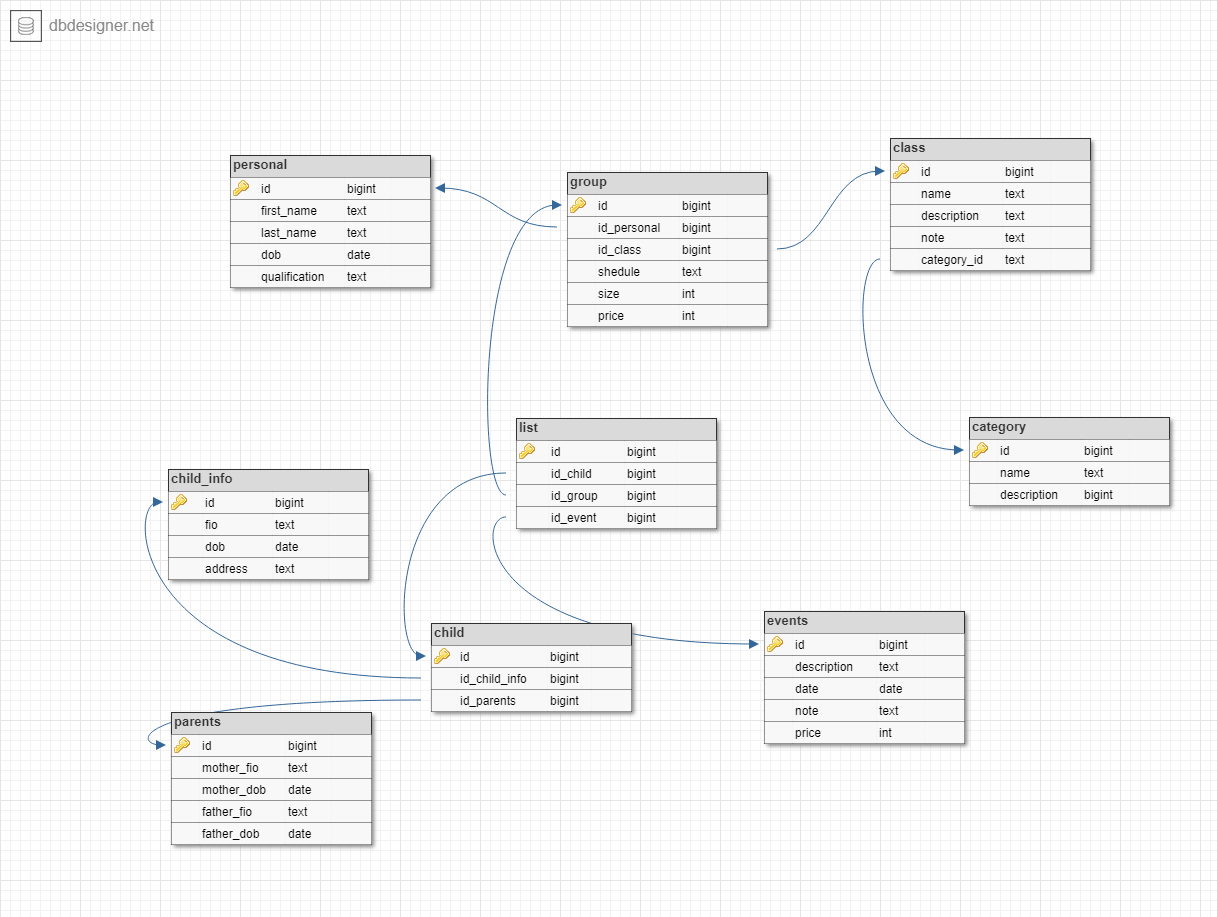


Рисунок 2.2.1 – ER диаграмма сущностей

Функциональные возможности:

* ведение БД (запись, чтение, модификация, удаление);
* обеспечение логической непротиворечивости БД;
* реализация наиболее часто встречающихся запросов в готовом виде;
* предоставление возможности сформировать произвольный запрос на языке манипулирования данными.

**З**апросы:

* получение списка всех групп, включая информацию о детях и классах;
* получение списка всех событий, включая информацию о детях;
* получение списка персонала;
* получение списка детей;
* получение списка полной информации о детях;
* получение списка родителей;
* получение списка классов;
* получение списка категорий;
* удаление данных из выше перечисленных списков;
* редактирование данных из выше перечисленных списков;
* добавление данных в выше перечисленные списки;

## 2.3. Обоснование выбора программных средств

СУБД MySql — предоставляет мощные средства для доступа, настройки, администрирования, разработки всех компонентов базы данных и управления ими. MySql — это реляционная система управления базами данных. То есть данные в ее базах хранятся в виде логически связанных между собой таблиц, доступ к которым осуществляется с помощью языка запросов SQL. MySql — свободно распространяемая система. Кроме того, это достаточно быстрая, надежная и, главное, простая в использовании СУБД.

Работать с MySql можно не только в текстовом режиме, но и в графическом. Существует очень популярный визуальный интерфейс для работы с этой СУБД — PhpMyAdmin. Этот интерфейс позволяет значительно упростить работу с базами данных в MySql.

PhpMyAdmin позволяет пользоваться всеми достоинствами браузера, включая прокрутку изображения, если оно не умещается на экран. Многие из базовых SQL-функций работы с данными в PhpMyAdmin сведены к интуитивно понятным интерфейсам и действиям, напоминающим переход по ссылкам в Internet.

Microsoft Visual Studio — линейка продуктов компании Майкрософт, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом.

Visual Studio включает в себя редактор исходного кода с поддержкой технологии IntelliSense и возможностью простейшего рефакторинга кода. Встроенный отладчик может работать как отладчик уровня исходного кода, так и как отладчик машинного уровня. Остальные встраиваемые инструменты включают в себя редактор форм для упрощения создания графического интерфейса приложения, веб-редактор, дизайнер классов и дизайнер схемы базы данных. VisualStudio позволяет создавать и подключать сторонние дополнения (плагины) для расширения функциональности практически на каждом уровне, включая добавление поддержки систем контроля версий исходного кода (например, Subversion и VisualSourceSafe), добавление новых наборов инструментов (например, для редактирования и визуального проектирования кода).

Главным преимуществом Visual Studio 2017 является производительность. Обеспечивает возможность создания разнообразных приложений на основе одного набора навыков .

C# — объектно-ориентированный язык программирования.

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.

Основные достоинства С#:

* C# создавался параллельно с каркасом Framework .Net и в полной мере учитывает все его возможности — как FCL, так и CLR;
* C# является полностью объектно-ориентированным языком, где даже типы, встроенные в язык, представлены классами;
* C# является мощным объектным языком с возможностями наследования и универсализации;
* C# является наследником языков C/C++, сохраняя лучшие черты этих популярных языков программирования. Общий с этими языками синтаксис, знакомые операторы языка облегчают переход программистов от С++ к C#;
* сохранив основные черты своего великого родителя, язык стал проще и надежнее. Простота и надежность, главным образом, связаны с тем, что на C# хотя и допускаются, но не поощряются такие опасные свойства С++ как указатели, адресация, разыменование, адресная арифметика;
* благодаря каркасу Framework .Net, ставшему надстройкой над операционной системой, программисты C# получают те же преимущества работы с виртуальной машиной, что и программисты Java. Эффективность кода даже повышается, поскольку исполнительная среда CLR представляет собой компилятор промежуточного языка, в то время как виртуальная Java-машина является интерпретатором байт-кода;
* мощная библиотека каркаса поддерживает удобство построения различных типов приложений на C#, позволяя легко строить Web-службы, другие виды компонентов, достаточно просто сохранять и получать информацию из базы данных и других хранилищ данных;
* реализация, сочетающая построение надежного и эффективного кода, является немаловажным фактором, способствующим успеху C#

## 2.4. Даталогическое проектирование

Даталогическое проектирование является проектированием логической структуры БД, что означает определение всех информационных единиц и связей между ними, задание их имен и типов, а также некоторых количественных характеристик (например, длины поля).

При проектировании логической структуры БД, осуществляется преобразование исходной инфологической модели в модель данных, поддерживаемую конкретной СУБД, и проверка адекватности, полученной даталогической модели отображаемой предметной области.

В данном курсовом проекте БД содержит 9 таблиц.

Таблица 1. Реляционная таблица «Categoty»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID категории |
| name | text |  | Название |
| description | text |  | Описание |

Таблица 2. Реляционная таблица «Child»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| id\_child\_info | bigint |  | id информации о ребёнке |
| id\_parents | bigint |  | id родителей |

Таблица 3. Реляционная таблица «Child\_info»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| fio | text |  | фио |
| dob | date |  | дата рождения |
| address | text |  | адрес |

  Реляционная таблица «Class»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| name | text |  | названиие |
| description | text |  | описание |
| note | text |  | заметки |
| category\_id | text |  | номер категории |

Таблица 5.Реляционная таблица «Events»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| description | text |  | описание |
| date | date |  | дата проведения |
| note | text |  | заметки |
| price | int |  | стоимость |

Таблица 6. Реляционная таблица «Group»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| id\_personal | bigint |  | учитель |
| id\_class | bigint |  | тип группы |
| shedule | text |  | распиание |
| size | int |  | размер группы |
| price | int |  | стоимость |

Таблица 7. Реляционная таблица «List»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| id\_child | bigint |  | id ребёнка |
| id\_group | bigint |  | id группы |
| id\_event | bigint |  | id события |

Таблица 8. Реляционная таблица «Parents»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| mother\_fio | text |  | фио матери |
| mother\_dob | date |  | дата рождения матери |
| father\_fio | text |  | фио отца |
| father\_dob | date |  | дата рождения отца |

Таблица 7. Реляционная таблица «Personal»

|  |  |  |  |
| --- | --- | --- | --- |
| Атрибуты | Тип | Ключ | Описание |
| id | bigint | primary | ID |
| fio | text |  | фио |
| dob | date |  | дата рождения |
| qualification | text |  | квалификация |

Таблицы соответствуют трём нормальным формам. Для связи между разными таблицами используется один и тот же ключ в разных таблицах и специально связующие таблицы. Все таблицы идентифицируются числом.

## 2.5. Физическое проектирование

Физическое проектирование базы данных - процесс подготовки описания реализации базы данных на вторичных запоминающих устройствах; на этом этапе рассматриваются основные отношения, организация файлов и индексов, предназначенных для обеспечения эффективного доступа к данным, а также все связанные с этим ограничения целостности и средства защиты.

Физическое проектирование является третьим и последним этапом создания проекта базы данных, при выполнении которого проектировщик принимает решения о способах реализации разрабатываемой базы данных. Приступая к физическому проектированию базы данных, прежде всего необходимо выбрать конкретную целевую СУБД. Поэтому физическое проектирование неразрывно связано с конкретной СУБД. Между логическим и физическим проектированием существует постоянная обратная связь, так как решения, принимаемые на этапе физического проектирования с целью повышения производительности системы, способны повлиять на структуру логической модели данных.

Как правило, основной целью физического проектирования базы данных является описание способа физической реализации логического проекта базы данных.

Для создания базы данных использовался следующий запрос:

CREATE TABLE `personal` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`first\_name` TEXT NOT NULL AUTO\_INCREMENT,

`last\_name` TEXT NOT NULL AUTO\_INCREMENT,

`dob` DATE NOT NULL AUTO\_INCREMENT,

`qualification` TEXT NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`id`)

);

CREATE TABLE `group` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`id\_personal` bigint NOT NULL AUTO\_INCREMENT,

`id\_class` bigint NOT NULL AUTO\_INCREMENT,

`shedule` TEXT NOT NULL,

`size` int NOT NULL,

`price` int NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `child` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`id\_child\_info` bigint NOT NULL AUTO\_INCREMENT,

`id\_parents` bigint NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (`id`)

);

CREATE TABLE `child\_info` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`fio` TEXT NOT NULL,

`dob` DATE NOT NULL,

`address` TEXT NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `class` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`name` TEXT NOT NULL,

`description` TEXT NOT NULL,

`note` TEXT NOT NULL,

`category\_id` TEXT NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `category` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`name` TEXT NOT NULL,

`description` bigint NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `parents` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`mother\_fio` TEXT NOT NULL,

`mother\_dob` DATE NOT NULL,

`father\_fio` TEXT NOT NULL,

`father\_dob` DATE NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `events` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`description` TEXT NOT NULL,

`date` DATE NOT NULL,

`note` TEXT NOT NULL,

`price` int NOT NULL,

PRIMARY KEY (`id`)

);

CREATE TABLE `list` (

`id` bigint NOT NULL AUTO\_INCREMENT,

`id\_child` bigint NOT NULL,

`id\_group` bigint NOT NULL,

`id\_event` bigint NOT NULL,

PRIMARY KEY (`id`)

);

ALTER TABLE `group` ADD CONSTRAINT `group\_fk0` FOREIGN KEY (`id\_personal`) REFERENCES `personal`(`id`);

ALTER TABLE `group` ADD CONSTRAINT `group\_fk1` FOREIGN KEY (`id\_class`) REFERENCES `class`(`id`);

ALTER TABLE `child` ADD CONSTRAINT `child\_fk0` FOREIGN KEY (`id\_child\_info`) REFERENCES `child\_info`(`id`);

ALTER TABLE `child` ADD CONSTRAINT `child\_fk1` FOREIGN KEY (`id\_parents`) REFERENCES `parents`(`id`);

ALTER TABLE `class` ADD CONSTRAINT `class\_fk0` FOREIGN KEY (`category\_id`) REFERENCES `category`(`id`);

ALTER TABLE `list` ADD CONSTRAINT `list\_fk0` FOREIGN KEY (`id\_child`) REFERENCES `child`(`id`);

ALTER TABLE `list` ADD CONSTRAINT `list\_fk1` FOREIGN KEY (`id\_group`) REFERENCES `group`(`id`);

ALTER TABLE `list` ADD CONSTRAINT `list\_fk2` FOREIGN KEY (`id\_event`) REFERENCES `events`(`id`);

Под физической организацией БД понимается совокупность методов и средств размещения данных во внешней памяти и созданная на их основе внутренняя (физическая) модель данных. Внутренняя модель является средством отображения логической модели данных в физическую среду хранения. В отличие от логических моделей физическая модель данных связана со способами организации данных на носителях, методами доступа к данным. Она указывает, каким образом записи размещаются в базе данных, как они упорядочиваются, как организуются связи, каким путем можно локализовать записи и осуществить их выборку.

Внутренняя модель разрабатывается средствами СУБД.

Очевидно, что любая логическая модель может быть отображена множеством внутренних моделей данных подобно тому, как один и тот же алгоритм может быть представлен множеством эквивалентных программ, составленных на одном или разных языках программирования. Одна из внутренних моделей будет оптимальной. В качестве критериев оптимальности используются минимальное время ответа системы, минимальный объем памяти, минимальные затраты на ведение баз данных и др.

Основными средствами физического моделирования в БнД являются структура хранения данных, поисковые структуры и язык описания данных.

В простейшем случае структуру хранения данных можно представить в виде структуры записи файла базы данных, включающей поля записи, порядок их размещения, типы и длины полей. Если структура хранения данных в основном предназначена для указания способа размещения записей и полей, то поисковые структуры определяют способ быстрого нахождения этих записей. Поэтому различают два принципа физической организации БД: организация на основе структуры хранения данных и организация, сочетающая структуру хранения данных с одной или несколькими поисковыми структурами.

Конечным итогом разработки физической организации БД являются базы данных – файл базы данных и файлы поисковых структур. В ПК эти файлы могут быть последовательными или прямыми (имеется в виду последовательного или прямого доступа).

В данном курсовом проекте все данные хранятся в таблицах БД и выводятся в виде таблиц в разработанной программе для работы с БД. Пример выводимых данных предоставлен на рисунке 2.5.1

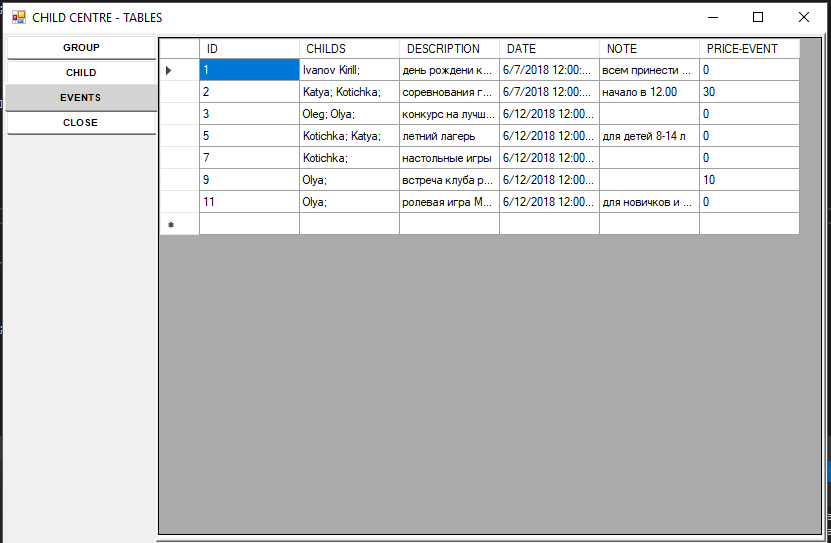


Рисунок 2.5.1. Вывод данных таблицы «Events»

## ЗАКЛЮЧЕНИЕ

Результатом выполнения курсовой работы стало разработанное приложение баз данных, позволяющее автоматизировать рутинные процессы по учету и управлению данными. в детских центрах. Разработанное приложение отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации. В данной работе разработана база данных и приложение для работы совместно с базой данных.

В процессе выполнения курсовой работы были закреплены знания, полученные при изучении дисциплины. Были изучены такие пункты:

* анализ предметной области;
* построение концептуальной модели базы данных;
* организация базы данных;
* разработка прикладной программы;
* наполнение и сопровождение базы данных;

В процессе организации БД проведен до необходимого уровня абстракций анализ предметной области, построена реляционная модель БД, произведена нормализация реляционной БД. Реализация проекта была выполнена на современных программных платформах. В качестве технологии доступа к данным была использована объектно-реляционная модель, которая позволяет просто и лаконично осуществлять запросы к базе данных.

Даная курсовая работа была реализована на языке программирования C# с использованием среды разработки Microsoft Visual Studio.

Для проверки корректности работы программы были изучены методы тестирования и проведены тесты, по результатам которых были исправлены ошибки.

Цель и соответствующие задачи, поставленные перед выполнение курсового проекта выполнены в полном объёме.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. В.А.Гвоздева, И.Ю.Лаврентьева. Основы построения АИС, Москва, ИД «Форум» - ИНФРА-М, 2009.

2. А.В.Рудаков, Технология разработки программных продуктов, Москва, Издательский центр «Академия», 2008

3. Л.Г.Гагарина, Д.В.Киселев, Е.Л.Федотова, Разработка и эксплуатация АИС. Москва, ИД «Форум» - ИНФРА-М, 2009.

4. Г.Ю.Максимович, А.Г.Романенко, О.Ф.Самойлюк. Информационные системы. Москва 2007, Федеральное агентство по образованию

5. Фаронов В.В. Программирование баз данных в Delphi 7: Учебный курс. – СПб.: Питер, 2004 - 464 с.

6. Кренке Д. Теория и практика построения баз данных: [пер.с англ] / Д. Кренке. - 9 - е изд. - СПб.: Питер, 2005. - 858 с.

## ПРИЛОЖЕНИЕ А

Листинг программы

Tables.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ChildCentre

{

public partial class Tables : Form

{

public Tables()

{

InitializeComponent();

}

private void tabControl1\_DrawItem(object sender, DrawItemEventArgs e)

{

Graphics g = e.Graphics;

Brush \_textBrush;

// Get the item from the collection.

TabPage \_tabPage = tabControl1.TabPages[e.Index];

// Get the real bounds for the tab rectangle.

Rectangle \_tabBounds = tabControl1.GetTabRect(e.Index);

if (e.State == DrawItemState.Selected)

{

// Draw a different background color, and don't paint a focus rectangle.

\_textBrush = new SolidBrush(Color.Black);

g.FillRectangle(Brushes.LightGray, e.Bounds);

}

else

{

\_textBrush = new System.Drawing.SolidBrush(e.ForeColor);

e.DrawBackground();

}

// Use our own font.

Font \_tabFont = new Font("Arial", (float)10.0, FontStyle.Bold, GraphicsUnit.Pixel);

// Draw string. Center the text.

StringFormat \_stringFlags = new StringFormat();

\_stringFlags.Alignment = StringAlignment.Center;

\_stringFlags.LineAlignment = StringAlignment.Center;

g.DrawString(\_tabPage.Text, \_tabFont, \_textBrush, \_tabBounds, new StringFormat(\_stringFlags));

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView2\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

TabPage tab = (sender as TabControl).SelectedTab;

Main main = new Main();

Console.WriteLine(tab.Name);

if (tab.Name == "close")

{

this.Hide();

main.Show();

}

}

private void Tables\_Load(object sender, EventArgs e)

{

SetupEventTable();

}

private void SetupEventTable()

{

List<Event> events = Event.getEvents();

foreach (Event item in events)

{

this.dataGridView3.Rows.Add(item.id, item.getChildsName(), item.description, item.date, item.note, item.price);

}

}

}

}

Admin.cs

using System;

using System.Collections.Generic;

using System.Drawing;

using System.Linq;

using System.Windows.Forms;

namespace ChildCentre

{

public partial class Admin : Form

{

Int32 eventCounter = 0;

Int32 childCounter = 0;

Int32 childInfoCounter = 0;

Int32 classCounter = 0;

Int32 groupCounter = 0;

Int32 listCounter = 0;

Int32 parentsCounter = 0;

Int32 personalCounter = 0;

public Admin()

{

InitializeComponent();

}

private void tabControl1\_DrawItem(object sender, DrawItemEventArgs e)

{

Graphics g = e.Graphics;

Brush \_textBrush;

// Get the item from the collection.

TabPage \_tabPage = tabControl1.TabPages[e.Index];

// Get the real bounds for the tab rectangle.

Rectangle \_tabBounds = tabControl1.GetTabRect(e.Index);

if (e.State == DrawItemState.Selected)

{

// Draw a different background color, and don't paint a focus rectangle.

\_textBrush = new SolidBrush(Color.Black);

g.FillRectangle(Brushes.LightGray, e.Bounds);

}

else

{

\_textBrush = new System.Drawing.SolidBrush(e.ForeColor);

e.DrawBackground();

}

// Use our own font.

Font \_tabFont = new Font("Arial", (float)10.0, FontStyle.Bold, GraphicsUnit.Pixel);

// Draw string. Center the text.

StringFormat \_stringFlags = new StringFormat();

\_stringFlags.Alignment = StringAlignment.Center;

\_stringFlags.LineAlignment = StringAlignment.Center;

g.DrawString(\_tabPage.Text, \_tabFont, \_textBrush, \_tabBounds, new StringFormat(\_stringFlags));

}

private void tabControl1\_SelectedIndexChanged(object sender, EventArgs e)

{

TabPage tab = (sender as TabControl).SelectedTab;

Main main = new Main();

Console.WriteLine(tab.Name);

if (tab.Name == "close")

{

this.Hide();

main.Show();

}

}

private void dataGridView3\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void Admin\_Load(object sender, EventArgs e)

{

SetupEventTable();

SetupCategoryTable();

SetupChildTable();

SetupChildInfoTable();

SetupParentsTable();

SetupPersonalTable();

SetupChildTable();

SetupClassTable();

SetupGroupTable();

SetupListTable();

}

private void SetupEventTable()

{

List<Event> events = Event.getEvents();

foreach (Event item in events)

{

this.dataGridView3.Rows.Add(item.id, item.description, item.date, item.note, item.price, "X");

}

eventCounter = 0;

}

private void SetupCategoryTable()

{

List<Category> categories = Category.getCategories();

foreach (Category item in categories)

{

this.dataGridView4.Rows.Add(item.id, item.name, item.description, "X");

}

}

private void SetupChildInfoTable()

{

List<ChildInfo> childInfos = ChildInfo.getChildInfos();

foreach (ChildInfo item in childInfos)

{

this.dataGridView5.Rows.Add(item.id, item.fio, item.dob, item.address, "X");

}

}

private void SetupParentsTable()

{

List<Parents> parents = Parents.getParents();

foreach (Parents item in parents)

{

this.dataGridView8.Rows.Add(item.id, item.mother\_fio, item.mother\_dob, item.father\_fio, item.father\_dob, "X");

}

}

private void SetupPersonalTable()

{

List<Personal> personals = Personal.getPersonal();

foreach (Personal item in personals)

{

this.dataGridView9.Rows.Add(item.id, item.fio, item.dob, item.qualification, "X");

}

}

private void SetupChildTable()

{

List<Child> childs = Child.getChild();

foreach (Child item in childs)

{

this.dataGridView2.Rows.Add(item.id, item.id\_child\_info, item.id\_parents, "X");

}

childCounter = 0;

}

private void saveChild(object sender, EventArgs e)

{

DataGridView childsTable = sender as DataGridView;

foreach (Int32 i in Enumerable.Range(1, childCounter))

{

var row = dataGridView2.Rows[dataGridView2.RowCount - i - 1].Cells;

Child newChild = new Child();

if (row[0].Value != null)

newChild.id = Convert.ToInt32(row[0].Value);

if (row[1].Value != null)

newChild.id\_child\_info = Convert.ToInt32(row[1].Value);

if (row[2].Value != null)

newChild.id\_parents = Convert.ToInt32(row[2].Value);

newChild.AddInDB();

}

}

private void SetupClassTable()

{

List<Class> classes = Class.getClass();

foreach (Class item in classes)

{

this.dataGridView6.Rows.Add(item.id, item.name, item.description, item.note, item.category\_id, "X");

}

}

private void SetupGroupTable()

{

List<Group> groups = Group.getGroups();

foreach (Group item in groups)

{

this.dataGridView1.Rows.Add(item.id, item.id\_personal,item.id\_class, item.shedule, item.size, item.price, "X");

}

}

private void SetupListTable()

{

List<List> lists = List.getLists();

foreach (List item in lists)

{

this.dataGridView7.Rows.Add(item.id, item.id\_child, item.id\_group, item.id\_event, "X");

}

}

private void dataGridView3\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

eventCounter++;

}

private void dataGridView3\_RowValidated(object sender, DataGridViewCellEventArgs e)

{

}

private void saveEvent(object sender, EventArgs e)

{

DataGridView eventsTable = sender as DataGridView;

foreach (Int32 i in Enumerable.Range(1, eventCounter))

{

var row = dataGridView3.Rows[dataGridView3.RowCount - i - 1].Cells;

Event newEvent = new Event();

if (row[0].Value != null)

newEvent.id = Convert.ToInt32(row[0].Value);

if (row[1].Value != null)

newEvent.description = row[1].Value.ToString();

if (row[2].Value != null)

newEvent.date = Convert.ToDateTime(row[2].Value);

if (row[3].Value != null)

newEvent.note = row[3].Value.ToString();

if (row[4].Value != null)

newEvent.price = Convert.ToInt32(row[4].Value);

newEvent.AddInDB();

}

}

private void dataGridView3\_CellClick(object sender, DataGridViewCellEventArgs e)

{

var cell = dataGridView3.Rows[e.RowIndex].Cells[e.ColumnIndex];

if (cell.Value != null)

{

if (cell.Value.ToString() == "X")

{

var id = dataGridView4.Rows[e.RowIndex].Cells[0].Value.ToString();

Event.RemoveFromID(id);

}

}

}

private void dataGridView4\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView4\_CellClick(object sender, DataGridViewCellEventArgs e)

{

var cell = dataGridView4.Rows[e.RowIndex].Cells[e.ColumnIndex];

if (cell.Value != null)

{

if (cell.Value.ToString() == "X")

{

var id = dataGridView4.Rows[e.RowIndex].Cells[0].Value.ToString();

Category.RemoveFromID(id);

}

}

}

private void dataGridView4\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

}

private void dataGridView2\_CellClick(object sender, DataGridViewCellEventArgs e)

{

}

private void dataGridView2\_RowsAdded(object sender, DataGridViewRowsAddedEventArgs e)

{

childCounter++;

}

}

}

Main.cs

using MySql.Data.MySqlClient;

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace ChildCentre

{

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

Application.Exit();

}

//private DBConnection dbconnect;

//public DBConnection DBConnect(

//{

// get

// {

// dbconnect = DBConnection.Instance();

// dbconnect.DatabaseName = "ChildCentre";

// return dbconnect;

// }

//}

private void button1\_Click(object sender, EventArgs e)

{

Tables table = new Tables();

table.Show();

this.Hide();

}

private void button2\_Click(object sender, EventArgs e)

{

Admin admin = new Admin();

admin.Show();

this.Hide();

}

}

}