**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ АСТРАХАНСКОЙ ОБЛАСТИ**

**Государственное бюджетное профессиональное образовательное учреждение  
Астраханской области «Астраханский колледж вычислительной техники»**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **К защите допустить:** | | | | | | | |
| **Старший методист УМС** | | | | | | | |
|  |  | | | **Л.О.Земцова** | | | |
| **«** |  | **»** |  | | **20** |  | **г.** |

**курсовая работа**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| по МДК | 04.01 | Технология разработки и защиты баз данных | | | | | | |
|  | | | | | | | | |
|  | | | 6 семестр | | | | | |
| Тема курсовой работы | | | Разработка базы данных | | | | | |
|  | | | ***«***Автошкола» | | | | | |
|  | | |  | | | | | |
| АКВТ.09.02.07.КР00.00ПЗ | | | | | | | | |
| Листов: | | | | 37 |  | | | |
|  | |  | | | | | | |
|  | |  | | | | | | |
| Выполнил | |  | | | | Студент гр. | | ПБ – 31 |
|  | |  | | | |  | | |
|  | |  | | | |  |  | |
| Проверил | |  | | | | Преподаватель | | |
|  | |  | | | | Мулеев Р.Р. | | |

**2022**

**Государственное бюджетное профессиональное образовательное учреждение**

**Астраханской области «Астраханский колледж вычислительной техники»**

|  |
| --- |
| УТВЕРЖДЕНО |
| Цикловой комиссией |
| специальности 09.02.07 |
| ПЦК \_\_\_\_\_\_\_\_\_ Ю.С. Андрианова |
| «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2022 г. |
| Специальность | *09.02.07 Информационные системы и программирование* | | |
| МДК 04.01 | *Технология разработки и защиты баз данных* | | |
| Группа | *ПБ– 31* | | |
| **ЗАДАНИЕ** | | | |
| **на курсовое проектирование** | | | |
| ***[ДАННЫЕ УДАЛЕНЫ]*** | | | |
| (фамилия, имя, отчество студента) | | | |
| Тема курсового проекта | | ***Разработка БД*** | |
| ***«Автошкола»*** | | | |

Содержание пояснительной записки курсового проекта

Содержание

Введение

1. Проектирование базы данных
2. Анализ и описание предметной области информационной системы
3. Проектирование концептуальной модели
4. Построение логической модели, схемы базы данных
5. Выбор СУБД
6. Реализация базы данных
7. Физическое проектирование
8. Написание исходного кода БД
9. Написание запросов, функций и хранимых процедур

Заключение

Список использованных источников

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | Р.Р. Мулеев |
| Студент |  |  |  |

**Аннотация**

Данный документ на разработку базы данных имеет 37 страницы, 4 раздела, 3 рисунка, 4 источника использованной литературы. Таблицы и приложения отсутствуют.

**Содержание**

[Введение 5](#_Toc106031958)

[1. Проектирование базы данных 6](#_Toc106031959)

[1.1 Анализ и описание предметной области информационной системы 6](#_Toc106031960)

[1.2 Проектирование концептуальной модели 7](#_Toc106031961)

[1.3 Построение логической модели, схемы базы данных 8](#_Toc106031962)

[1.4 Выбор СУБД 14](#_Toc106031963)

[2 Реализация базы данных 16](#_Toc106031964)

[2.1 Физическое проектирование БД 16](#_Toc106031965)

[2.2 Написание исходного кода БД 16](#_Toc106031966)

[2.3 Написание запросов, функций и хранимых процедур 21](#_Toc106031967)

[Заключение 36](#_Toc106031968)

[Список используемой литературы 37](#_Toc106031969)

# Введение

Современные условия для работы по обслуживанию достигло того уровня, когда весь документооборот осуществляется на компьютерах, и вся информация храниться в специальных базах, потому растёт значимость информационных систем, которые позволяют обеспечить информационную поддержку процессов принятия решений. Базы данных являются одним из основных элементов большинства информационных систем.

База данных (далее БД) - это организованная структура, в которой в специальном формате хранится информация.

Система управления базой данных (далее СУБД) - это программа, с помощью которой в компьютер вводится информация, просматривается, сортируется и т.д.

БД «Автошкола» предназначена для структуризации информации о заявках в автошколу, учеников, учителей и категориях и мастеров.

# Проектирование базы данных

## Анализ и описание предметной области информационной системы

Автошкола – образовательное учреждение, в котором происходит обучение вождению транспортных средств.

Целью спортивного клуба является:

* повысить уровень грамотности населения в сфере управления дорожным транспортом;
* получение прибыли от оказания услуг образовательного характера.

Предметная область представляет собой базу данных, направлена на информационную поддержку автошкола. База данных даёт возможность проведение аналитики и создание отчётности по деятельности всей автошколы, а также каждого ученика, автоматизирует некоторые процессы. Ввиду всех возможностей базы данных возникает необходимость в автоматизации информационных процессов.

Входные документы:

* заявка на поступление в автошколу;
* категории и экзамены к ним.

Выходные документы:

* список оплативших обучение;
* список курсантов (по группам и сводный);
* лист регистрации инструктажа;
* результаты внутришкольного экзамена;
* свидетельство об окончании автошколы.

База данных создана для руководителей автошколы, которые могут просматривать, вносить, изменять, удалять информацию в базе данных в полном объёме.

БД должна осуществлять:

* ведение справочника потенциальных учеников автошколы, на основе заполняемой ими через интернет-страницу заявки на обучение;
* ведение справочника учащихся с разбивкой на группы по категориям и указанием периода обучения;
* ведение справочника преподавателей и мастеров с закреплением их по группам;
* контроль оплаты обучения;
* фиксацию результатов экзаменов.

## Проектирование концептуальной модели

Сущности:

1. Результаты экзамена - содержит информацию о результатах экзамена.

Атрибуты: результат.

1. Заявки - содержит информацию о заявках в автошколу.

Атрибуты: ФИО, номер телефона.

1. Ученики - содержит информацию о учениках.

Атрибуты: дата принятия в автошколу, оплата.

1. Группы - содержит информацию о группах.

Атрибуты: категория, название.

1. Учителя - содержит информацию о учителях.

Атрибуты: ФИО, номер телефона.

1. Мастера - содержит информацию о мастерах.

Атрибуты: ФИО, номер телефона.

Связи:

1. Заявка принимается в ученики. Одна заявка может приняться только один раз (связь 1 к 1).
2. Ученик сдаёт в экзамен. Ученик может сдать несколько экзаменов, а экзамен сдаёт один ученик (связь 1 к многим).
3. Ученик учится в группе. В группе может учится много учеников, а ученик может учиться только в одной группе (связь 1 к многим).
4. Мастер откосится к группе. К группе может относится много мастеров, а у мастера может быть несколько групп (связь много к многим).
5. Учитель откосится к группе. К группе может относится много учителей, а у учителя может быть несколько групп (связь много к многим).

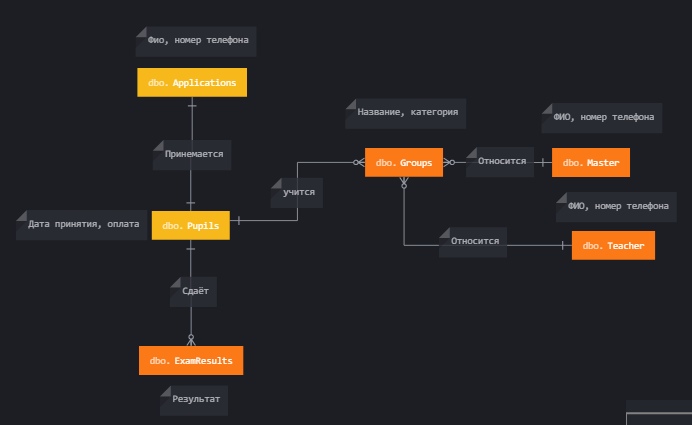


Рисунок 1 – Концептуальная модель

## Построение логической модели, схемы базы данных

1. Заявка (Applications):
   1. Идентификатор заявки (ApplicationsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. Номер телефона (PhoneNumber):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: международный стандарт телефонных номеров.
  1. День рождения (Birthday):
* нулевое значение: not null;
* тип поля: date.

1. Ученики (Pupils):
   1. Идентификатор ученика (PupilsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Дата поступления (DateAdoption):
* нулевое значение: not null;
* тип поля: date.
  1. Идентификатор заявки (ApplicationsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Applications (поле ApplicationsID).

1. Группы учеников (PupilsGroups):
   1. Идентификатор группы ученика (PupilsGroupsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Дата начала обучения (DateStart):
* нулевое значение: not null;
* тип поля: date.
  1. Идентификатор ученика (PupilsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Pupils (поле PupilsID).
  1. Идентификатор группы (GroupsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Groups(поле GroupsID).
  1. Дата окончания обучения (DateFinish):
* нулевое значение: not null;
* тип поля: date.

1. Результаты экзамена (ExamResults):
   1. Идентификатор результатов экзамена (ExamResultsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор ученика группы (PupilsGroupsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу PupilsGroups (поле PupilsGroupsID).
  1. Баллы (Score):
* нулевое значение: not null;
* тип поля: int.

1. Категория (Categories):
   1. Идентификатор категории (CategoriesID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название категории (CategoriesName):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Файл экзамена (ExamsFile):
* нулевое значение: not null;
* тип поля: varbinary.
  1. Максимальное количество баллов (MaxExamsScore):
* нулевое значение: not null;
* тип поля: int;
* маска ввода: больше 0.

1. Группы (Groups):
   1. Идентификатор клиента (GroupsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Название группы (GroupsName):
* нулевое значение: not null;
* тип поля: nvarchar.
  1. Идентификатор категории (CategoriesID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Categories (поле CategoriesID).

1. Учителя группы (TeacherGroups):
   1. Идентификатор учителя группы (TeacherGroupsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор группы (GroupsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Groups (поле GroupsID).
  1. Идентификатор учителя (TeacherID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Teacher(поле TeacherID).

1. Учитель (Teacher):
   1. Идентификатор учителя (TeacherID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. Номер телефона (PhoneNumber):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: международный стандарт телефонных номеров.

1. Учитель (Master):
   1. Идентификатор мастера (MasterID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Фамилия, имя, отчество (FirstName, MiddleName, LastName):
* нулевое значение: FirstName и MiddleName not null, LastName null;
* тип поля: nvarchar;
* нормализированная форма атрибута ФИО.
  1. Номер телефона (PhoneNumber):
* нулевое значение: not null;
* тип поля: bigint;
* маска ввода: международный стандарт телефонных номеров.

1. Мастера группы (MasterGroups):
   1. Идентификатор мастера группы (MasterGroupsID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор группы (GroupsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Groups (поле GroupsID).
  1. Идентификатор учителя (MasterID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу Master(поле MasterID).

1. Инструктаж (Briefing):
   1. Идентификатор игроков турнира (BriefingID):

* нулевое значение: not null;
* тип поля: int;
* первичный ключ, созданный искусственно из-за отсутствия атрибута идентификатора у сущности;
* автоинкримированный.
  1. Идентификатор мастера группы (MasterGroupsID):
* нулевое значение: not null;
* тип поля: int;
* вторичный ключ, ссылающийся на таблицу MasterGroups (поле MasterGroupsID).
  1. Дата начала (dateStart):
* нулевое значение: not null;
* тип поля: datetime.
  1. Дата окончания (dateFinish):
* нулевое значение: not null;
* тип поля: datetime.

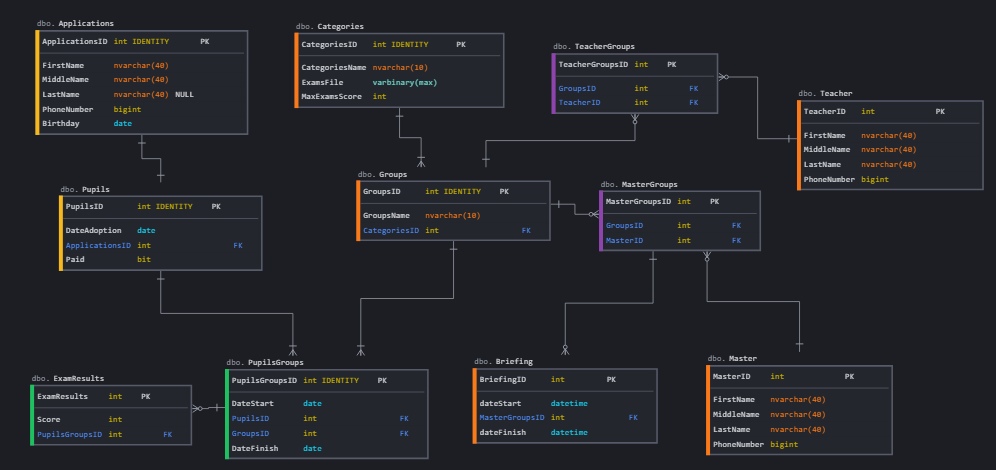


Рисунок 2 – Логическая модель

## Выбор СУБД

СУБД предназначена для централизованного управления базой данных в интересах, всех работающих в этой системе. Используемые в настоящее время СУБД, обладающих средствами обеспечения целостности данных и надёжной безопасности, что даёт возможность разработчикам гарантировать большую безопасность данных при меньших затратах сил на низкоуровневое программирование.

В качестве средства реализации БД была выбрана система Microsoft SQL Server Management Studio 2019 (далее MSSQL).

MSSQL– это мощная и надежная система управления данными, обеспечивающая множество функций, защиту данных и высокую производительность для внедренных приложений-клиентов и локальных хранилищ данных. MSSQL предназначен для упрощенного развертывания и быстрого создания прототипов; его можно получить бесплатно и свободно распространять вместе с приложениями. Также можно сказать, что он удовлетворяет требованиям, предъявляемым к системам распределенной обработки информации. Эта СУБД поддерживает: тиражирование данных, параллельную обработку, создание и обработку больших баз данных на недорогих аппаратных платформах.

MSSQL упрощает развёртывание, управление и оптимизацию данных предприятия и аналитических приложений. Как платформа управления данными предприятия, он предоставляет единую консоль управления, которая позволяет администраторам данных, находящимся в любом месте вашей организации, отслеживать, управлять и настраивать все базы данных и связанные службы по всему предприятию. Он предоставляет расширяемую инфраструктуру управления, которая может быть легко запрограммирована при помощи SQL Management Objects, позволяя пользователям переделывать и расширять их среду управления и независимым поставщикам программных продуктов создавать дополнительные инструменты и функциональность для дальнейшего расширения возможностей, поставляемых по умолчанию.

Средством, связывающим клиента с сервером, является язык SQL - язык структурированных запросов. В данном курсовом проекте были использованы основные операторы этого языка. Этот язык является очень распространенным на данный момент и имеет ряд особенностей.

# Реализация базы данных

## Физическое проектирование БД

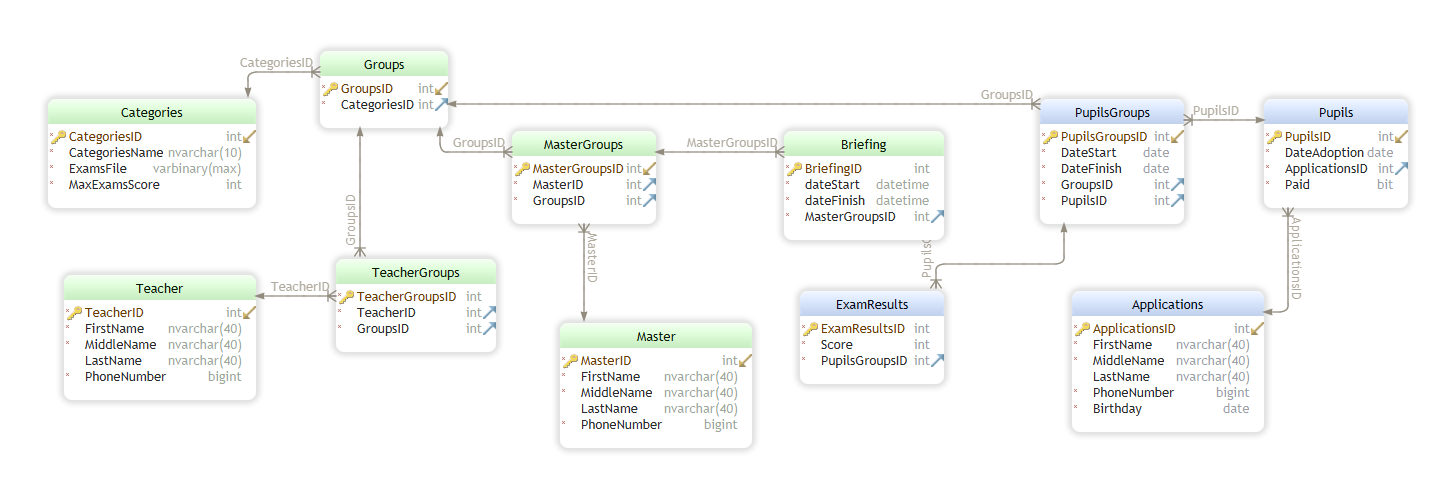


Рисунок 3 – Диаграмма базы данных

## Написание исходного кода БД

* 1. Applications

Данная таблица создана с целью хранения информации о заявках.

Листинг:

CREATE TABLE Applications (

ApplicationsID int NOT NULL IDENTITY(1, 1),

FirstName nvarchar(40) NOT NULL,

MiddleName nvarchar(40) NOT NULL,

LastName nvarchar(40) NULL,

PhoneNumber bigint NOT NULL,

Birthday date NOT NULL,

CONSTRAINT Applications\_pk PRIMARY KEY (ApplicationsID)

);

alter table

Applications ADD CONSTRAINT Applications\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10)

* 1. Briefing

Данная таблица создана с целью хранения информации о инструктажах.

Листинг:

CREATE TABLE Briefing (

BriefingID int NOT NULL IDENTITY(1, 1),

dateStart datetime NOT NULL,

dateFinish datetime NOT NULL,

MasterGroupsID int NOT NULL,

CONSTRAINT Briefing\_pk PRIMARY KEY (BriefingID)

);

ALTER TABLE Briefing ADD CONSTRAINT Briefing\_MasterGroups

FOREIGN KEY (MasterGroupsID)

REFERENCES MasterGroups (MasterGroupsID)

ON DELETE CASCADE

ON UPDATE CASCADE;

* 1. Categories

Данная справочная таблица создана с целью хранения информации о котегориях.

Листинг:

CREATE TABLE Categories (

CategoriesID int NOT NULL IDENTITY(1, 1),

CategoriesName nvarchar(10) NOT NULL,

ExamsFile varbinary(max) NOT NULL,

MaxExamsScore int NOT NULL,

CONSTRAINT check\_MaxExamsScore CHECK (MaxExamsScore > 0),

CONSTRAINT Categories\_pk PRIMARY KEY (CategoriesID)

);

* 1. ExamResults

Данная таблица создана с целью хранения информации о результатах экзамена

Листинг:

CREATE TABLE ExamResults (

ExamResultsID int NOT NULL IDENTITY(1, 1),

Score int NOT NULL,

PupilsGroupsID int NOT NULL,

CONSTRAINT check\_Score CHECK (Score >= 0),

CONSTRAINT ExamResults\_pk PRIMARY KEY (ExamResultsID)

);

ALTER TABLE ExamResults ADD CONSTRAINT ExamResults\_PupilsGroups

FOREIGN KEY (PupilsGroupsID)

REFERENCES PupilsGroups (PupilsGroupsID)

ON DELETE CASCADE

ON UPDATE CASCADE;

* 1. Groups

Данная таблица создана с целью хранения информации о группах.

Листинг:

CREATE TABLE Groups (

GroupsID int NOT NULL IDENTITY(1, 1),

CategoriesID int NOT NULL,

CONSTRAINT Groups\_pk PRIMARY KEY (GroupsID)

);

ALTER TABLE Groups ADD CONSTRAINT Groups\_Categories

FOREIGN KEY (CategoriesID)

REFERENCES Categories (CategoriesID)

ON DELETE CASCADE

ON UPDATE CASCADE;

* 1. Master

Данная таблица создана с целью хранения информации о мастерах.

Листинг:

CREATE TABLE Master (

MasterID int NOT NULL IDENTITY(1, 1),

FirstName nvarchar(40) NOT NULL,

MiddleName nvarchar(40) NOT NULL,

LastName nvarchar(40) NULL,

PhoneNumber bigint NOT NULL,

CONSTRAINT Master\_pk PRIMARY KEY (MasterID)

);

alter table

Master ADD CONSTRAINT Master\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10)

* 1. MasterGroups

Данная таблица создана с целью хранения информации о мастерах группах.

Листинг:

CREATE TABLE MasterGroups (

MasterGroupsID int NOT NULL IDENTITY(1, 1),

MasterID int NOT NULL,

GroupsID int NOT NULL,

CONSTRAINT MasterGroups\_pk PRIMARY KEY (MasterGroupsID)

);

ALTER TABLE MasterGroups ADD CONSTRAINT MasterGroups\_Groups

FOREIGN KEY (GroupsID)

REFERENCES Groups (GroupsID)

ON DELETE CASCADE

ON UPDATE CASCADE;

ALTER TABLE MasterGroups ADD CONSTRAINT MasterGroups\_Master

FOREIGN KEY (MasterID)

REFERENCES Master (MasterID);

* 1. Pupils

Данная таблица создана с целью хранения информации о учениках.

Листинг:

CREATE TABLE Pupils (

PupilsID int NOT NULL IDENTITY(1, 1),

DateAdoption date NOT NULL,

ApplicationsID int NOT NULL,

Paid bit NOT NULL,

CONSTRAINT Pupils\_pk PRIMARY KEY (PupilsID)

);

ALTER TABLE Pupils ADD CONSTRAINT Pupils\_Applications

FOREIGN KEY (ApplicationsID)

REFERENCES Applications (ApplicationsID)

ON DELETE CASCADE

ON UPDATE CASCADE;

* 1. PupilsGroups

Данная таблица создана с целью хранения информации о учеников группы.

Листинг:

CREATE TABLE PupilsGroups (

PupilsGroupsID int NOT NULL IDENTITY(1, 1),

DateStart date NOT NULL,

DateFinish date NOT NULL,

GroupsID int NOT NULL,

PupilsID int NOT NULL,

CONSTRAINT PupilsGroups\_pk PRIMARY KEY (PupilsGroupsID)

);

ALTER TABLE PupilsGroups ADD CONSTRAINT PupilsGroups\_Groups

FOREIGN KEY (GroupsID)

REFERENCES Groups (GroupsID);

ALTER TABLE PupilsGroups ADD CONSTRAINT PupilsGroups\_Pupils

FOREIGN KEY (PupilsID)

REFERENCES Pupils (PupilsID)

ON DELETE CASCADE

ON UPDATE CASCADE;

* 1. Teacher

Данная таблица создана с целью хранения информации о учителях.

Листинг:

CREATE TABLE Teacher (

TeacherID int NOT NULL IDENTITY(1, 1),

FirstName nvarchar(40) NOT NULL,

MiddleName nvarchar(40) NOT NULL,

LastName nvarchar(40) NULL,

PhoneNumber bigint NOT NULL,

CONSTRAINT Teacher\_pk PRIMARY KEY (TeacherID)

);

alter table

Teacher ADD CONSTRAINT Master\_check\_phoheNumber CHECK (DATALENGTH(CAST(PhoneNumber AS varchar(10))) = 10)

* 1. TeacherGroups

Данная таблица создана с целью хранения информации о учителях группах.

Листинг:

CREATE TABLE TeacherGroups (

TeacherGroupsID int NOT NULL IDENTITY(1, 1),

TeacherID int NOT NULL,

GroupsID int NOT NULL,

CONSTRAINT TeacherGroups\_pk PRIMARY KEY (TeacherGroupsID)

);

ALTER TABLE TeacherGroups ADD CONSTRAINT TeacherGroups\_Groups

FOREIGN KEY (GroupsID)

REFERENCES Groups (GroupsID);

ALTER TABLE TeacherGroups ADD CONSTRAINT TeacherGroups\_Teacher

FOREIGN KEY (TeacherID)

REFERENCES Teacher (TeacherID);

## Написание запросов, функций и хранимых процедур

1. ApplicationsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Applications и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Applications по названию и типу.

Листинг процедуры:

Create PROCEDURE ApplicationsAction (

@Action tinyint,

@ApplicationsID INT ,

@FirstName nvarchar(40),

@MiddleName nvarchar(40),

@LastName nvarchar(40),

@PhoneNumber bigint,

@Birthday date

)

As

Begin

if(@Action = 1)

delete Applications where ApplicationsID = @ApplicationsID

else

if(@Action = 2)

insert Applications (FirstName,MiddleName,LastName,PhoneNumber,Birthday)

values (@FirstName,@MiddleName,@LastName,@PhoneNumber,@Birthday)

else

if(@Action = 3)

UPDATE [dbo].[Applications]

SET FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

,Birthday = @Birthday

where ApplicationsID = @ApplicationsID

end

Пример использования:

* Удаление: execute ApplicationsAction 1,2,’’,’’,’’,null,null

При удалении важно указать первым (@Action) 1 и вторым (@ApplicationsID) существующий ApplicationsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute ApplicationsAction 2,1, ’wqe’, ’qwe’, ’qwe’,3624058956, ’04.04.2003’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ApplicationsAction 3,1, ’вав’,’sf’,’sfd’, 3624058956, ’04.04.2003’

При обновлении важно указать первым (@Action) 3 и вторым (@ApplicationsID) существующий ApplicationsID, а остальное заполнить нужными значениями.

1. BriefingAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Briefing и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Briefing по названию и типу.

Листинг процедуры:

Create PROCEDURE BriefingAction (

@Action tinyint,

@BriefingID INT,

@dateStart datetime,

@dateFinish datetime,

@MasterGroupsID int

)

As

Begin

if(@Action = 1)

delete Briefing where BriefingID = @BriefingID

else

if(@Action = 2)

insert Briefing (dateStart,dateFinish,MasterGroupsID)

values (@dateStart,@dateFinish,@MasterGroupsID)

else

if(@Action = 3)

UPDATE [dbo].[Briefing]

SET dateStart = @dateStart

,dateFinish = @dateFinish

,MasterGroupsID = @MasterGroupsID

where BriefingID = @BriefingID

end

Пример использования:

* Удаление: execute BriefingAction 1,2,’’,’’,0

При удалении важно указать первым (@Action) 1 и вторым (@BriefingID) существующий BriefingID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute BriefingAction 2,1, ’04.04.2013’,’05.05.2013’,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute BriefingAction 3,1, ’04.04.2013’,’05.05.2013’,1

При обновлении важно указать первым (@Action) 3 и вторым (@BriefingID) существующий BriefingID, а остальное заполнить нужными значениями.

1. CategoriesAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Categories и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Categories по названию и типу.

Листинг процедуры:

Create PROCEDURE CategoriesAction (

@Action tinyint,

@CategoriesID INT,

@CategoriesName nvarchar(10),

@ExamsFile varbinary(max),

@MaxExamsScore int

)

As

Begin

if(@Action = 1)

delete Categories where CategoriesID = @CategoriesID

else

if(@Action = 2)

insert Categories (CategoriesName,ExamsFile,MaxExamsScore)

values (@CategoriesName,@ExamsFile,@MaxExamsScore)

else

if(@Action = 3)

UPDATE [dbo].[Categories]

SET dateStart = @dateStart

,dateFinish = @dateFinish

,MasterGroupsID = @MasterGroupsID

where CategoriesID = @CategoriesID

end

Пример использования:

* Удаление: execute CategoriesAction 1,2,’’

При удалении важно указать первым (@Action) 1 и вторым (@CategoriesID) существующий CategoriesID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute CategoriesAction 2,1,’’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute CategoriesAction 3,1,’’

При обновлении важно указать первым (@Action) 3 и вторым (@CategoriesID) существующий CategoriesID, а остальное заполнить нужными значениями.

1. ExamResultsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу ExamResults и включает следующие переменные: @Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице ExamResults по названию и типу.

Листинг процедуры:

Create PROCEDURE ExamResultsAction (

@Action tinyint,

@ExamResultsID INT,

@Score int,

@PupilsGroupsID int

)

As

Begin

if(@Action = 1)

delete ExamResults where ExamResultsID = @ExamResultsID

else

if(@Action = 2)

insert ExamResults (Score,PupilsGroupsID)

values (@Score,@PupilsGroupsID)

else

if(@Action = 3)

UPDATE [dbo].[ExamResults]

SET Score = @Score

,PupilsGroupsID = @PupilsGroupsID

where ExamResultsID = @ExamResultsID

end

Пример использования:

* Удаление: execute ExamResultsAction 1,2,0,0

При удалении важно указать первым (@Action) 1 и вторым (@ExamResultsID) существующий ExamResultsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute ExamResultsAction 2,1, 100,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute ExamResultsAction 3,1, 11, 1,

При обновлении важно указать первым (@Action) 3 и вторым (@ExamResultsID) существующий ExamResultsID, а остальное заполнить нужными значениями.

1. GroupsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Groups и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Groups по названию и типу.

Листинг процедуры:

Create PROCEDURE GroupsAction (

@Action tinyint,

@GroupsID INT,

@CategoriesID int

)

As

Begin

if(@Action = 1)

delete Groups where GroupsID = @GroupsID

else

if(@Action = 2)

insert Groups (CategoriesID)

values (@CategoriesID)

else

if(@Action = 3)

UPDATE [dbo].[Groups]

SET CategoriesID = @CategoriesID

where GroupsID = @GroupsID

end

Пример использования:

* Удаление: execute GroupsAction 1,2,1,

При удалении важно указать первым (@Action) 1 и вторым (@GroupsID) существующий GroupsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute GroupsAction 2,1,2

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute GroupsAction 3, 1,1

При обновлении важно указать первым (@Action) 3 и вторым (@GroupsID) существующий GroupsID, а остальное заполнить нужными значениями.

1. MasterAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Master и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Master по названию и типу.

Листинг процедуры:

Create PROCEDURE MasterAction (

@Action tinyint,

@MasterID INT,

@FirstName nvarchar(40),

@MiddleName nvarchar(40),

@LastName nvarchar(40),

@PhoneNumber bigint

)

As

Begin

if(@Action = 1)

delete Master where MasterID = @MasterID

else

if(@Action = 2)

insert Master (FirstName,MiddleName,LastName,PhoneNumber)

values (@FirstName,@MiddleName,@LastName,@PhoneNumber)

else

if(@Action = 3)

UPDATE [dbo].[Master]

SET FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

where MasterID = @MasterID

end

Пример использования:

* Удаление: execute MasterAction 1,2, ’wqe’, ’qwe’, ’qwe’,3624058956, ’04.04.2003’

При удалении важно указать первым (@Action) 1 и вторым (@MasterID) существующий MasterID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute MasterAction 2,1, ’wqe’, ’qwe’, ’qwe’,3624058956, ’04.04.2003’

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute MasterAction 3,1, ’wqe’, ’qwe’, ’qwe’,3624058956, ’04.04.2003’

При обновлении важно указать первым (@Action) 3 и вторым (@MasterID) существующий MasterID, а остальное заполнить нужными значениями.

1. MasterGroupsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу MasterGroups и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице MasterGroups по названию и типу.

Листинг процедуры:

Create PROCEDURE MasterGroupsAction (

@Action tinyint,

@MasterGroupsID INT,

@MasterID int,

@GroupsID int

)

As

Begin

if(@Action = 1)

delete MasterGroups where MasterGroupsID = @MasterGroupsID

else

if(@Action = 2)

insert MasterGroups (MasterID,GroupsID )

values (@MasterID,@GroupsID )

else

if(@Action = 3)

UPDATE [dbo].[MasterGroups]

SET MasterID = @MasterID

,GroupsID = @GroupsID

where MasterGroupsID = @MasterGroupsID

end

Пример использования:

* Удаление: execute MasterGroupsAction 1,1,1,1

При удалении важно указать первым (@Action) 1 и вторым (@MasterGroupsID) существующий MasterGroupsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute MasterGroupsAction 2, 1,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute MasterGroupsAction 3, 1,1,1

При обновлении важно указать первым (@Action) 3 и вторым (@MasterGroupsID) существующий MasterGroupsID, а остальное заполнить нужными значениями.

1. PupilsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Pupils и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Pupils по названию и типу.

Листинг процедуры:

Create PROCEDURE PupilsAction (

@Action tinyint,

@PupilsID INT,

@DateAdoption date,

@ApplicationsID int,

@Paid bit

)

As

Begin

if(@Action = 1)

delete Pupils where PupilsID = @PupilsID

else

if(@Action = 2)

insert Pupils (DateAdoption,ApplicationsID,Paid)

values (@DateAdoption,@ApplicationsID,@Paid)

else

if(@Action = 3)

UPDATE [dbo].[Pupils]

SET DateAdoption = @DateAdoption

,ApplicationsID = @ApplicationsID

,Paid = @Paid

where PupilsID = @PupilsID

end

Пример использования:

* Удаление: execute PupilsAction 1,2,’01.01.2012’,1,1

При удалении важно указать первым (@Action) 1 и вторым (@PupilsID) существующий PupilsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute PupilsAction 2, 1, ’01.01.2012’,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute PupilsAction 3,1, ’01.01.2012’,1,1

При обновлении важно указать первым (@Action) 3 и вторым (@PupilsID) существующий PupilsID, а остальное заполнить нужными значениями.

1. PupilsGroupsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу PupilsGroups и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице PupilsGroups по названию и типу.

Листинг:

Create PROCEDURE PupilsGroupsAction (

@Action tinyint,

@PupilsGroupsID INT,

@DateStart date,

@DateFinish date,

@GroupsID int,

@PupilsID int

)

As

Begin

if(@Action = 1)

delete PupilsGroups where PupilsGroupsID = @PupilsGroupsID

else

if(@Action = 2)

insert PupilsGroups (DateStart,DateFinish,GroupsID,PupilsID)

values (@DateStart,@DateFinish,@GroupsID,@PupilsID)

else

if(@Action = 3)

UPDATE [dbo].[PupilsGroups]

SET DateStart = @DateStart

,DateFinish = @DateFinish

,GroupsID = @GroupsID

,PupilsID = @PupilsID

where PupilsGroupsID = @PupilsGroupsID

end

Пример использования:

* Удаление: execute PupilsGroupsAction 1,1,’12.12.2012’, ’12.12.2015’,1,1

При удалении важно указать первым (@Action) 1 и вторым (@PupilsGroupsID) существующий PupilsGroupsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute PupilsGroupsAction 2, 1, ’12.12.2012’, ’12.12.2015’,1,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute PupilsGroupsAction 3, 1, ’12.12.2012’, ’12.12.2015’,1,1

При обновлении важно указать первым (@Action) 3 и вторым (@PupilsGroupsID) существующий PupilsGroupsID, а остальное заполнить нужными значениями.

1. TeacherAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу Teacher и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице Teacher по названию и типу.

Листинг процедуры:

Create PROCEDURE TeacherAction (

@Action tinyint,

@TeacherID INT,

@FirstName nvarchar(40),

@MiddleName nvarchar(40),

@LastName nvarchar(40),

@PhoneNumber bigint

)

As

Begin

if(@Action = 1)

delete Teacher where TeacherID = @TeacherID

else

if(@Action = 2)

insert Teacher (FirstName,MiddleName,LastName,PhoneNumber)

values (@FirstName,@MiddleName,@LastName,@PhoneNumber)

else

if(@Action = 3)

UPDATE [dbo].[Teacher]

SET FirstName = @FirstName

,MiddleName = @MiddleName

,LastName = @LastName

,PhoneNumber = @PhoneNumber

where TeacherID = @TeacherID

end

Пример использования:

* Удаление: execute TeacherAction 1,1,null,0,null,null,0

При удалении важно указать первым (@Action) 1 и вторым (@TeacherID) существующий TeacherID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TeacherAction 2, 1, ‘esseg’,10,’12.12.2003’, ’22.12.2003’,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TeacherAction 3, 1, ‘esseg’,10,’12.12.2003’, ’22.12.2003’,1

При обновлении важно указать первым (@Action) 3 и вторым (@TeacherID) существующий TeacherID, а остальное заполнить нужными значениями.

1. TeacherGroupsAction

Описание: процедура создана для вставки, удаления, обновления строк в таблицу TeacherGroups и включает следующие переменные: @ Action (1 – удаление, 2 – добавление, 3 – обновление) и переменные, соответствующие полям в таблице TeacherGroups по названию и типу.

Листинг процедуры:

Create PROCEDURE TeacherGroupsAction (

@Action tinyint,

@TeacherGroupsID INT,

@TeacherID int,

@GroupsID int

)

As

Begin

if(@Action = 1)

delete TeacherGroups where TeacherGroupsID = @TeacherGroupsID

else

if(@Action = 2)

insert TeacherGroups (TeacherID,GroupsID)

values (@TeacherID,@GroupsID)

else

if(@Action = 3)

UPDATE [dbo].[TeacherGroups]

SET TeacherID = @TeacherID

,GroupsID = @GroupsID

where TeacherGroupsID = @TeacherGroupsID

end

Пример использования:

* Удаление: execute TeacherGroupsAction 1,1,null,null,

При удалении важно указать первым (@Action) 1 и вторым (@TeacherGroupsID) существующий TeacherGroupsID, остальные значения не важны, их можно заменить кавычками.

* Добавление: execute TeacherGroupsAction 2, 1,1 ,1

При добавлении важно указать первым (@Action) 2, а остальное заполнить нужными значениями.

* Обновление: execute TeacherGroupsAction 3, 1,1 ,1

При обновлении важно указать первым (@Action) 3 и вторым (@TeacherGroupsID) существующий TeacherGroupsID, а остальное заполнить нужными значениями.

1. PupilsPaid

Описание: процедура создана для вывода оплативших учеников.

Листинг:

Create PROCEDURE PupilsPaid

As

Begin

select \* from Pupils

where Paid = 1

end

Выходные данные: все поля таблицы Pupils.

1. EventInfo

Описание: процедура создана для вывода информации о учениках по группам.

Листинг:

Create PROCEDURE PupilsInfo

As

Begin

select p.\*, g.\* from Pupils p

inner join PupilsGroups pg on pg.PupilsID = p.PupilsID

inner join Groups g on g.GroupsID = pg.GroupsID

end

Выходные данные: все поля таблицы Pupils, все поля таблицы Groups.

1. checkInsertPupilsGroups

Триггер создан как ограничение таблицы PupilsGroups на вставку и обновление полей с условием: дата начала партии не может больше даты окончания партии.

Листинг:

create trigger checkInsertPupilsGroups on PupilsGroups

for insert, update

as

declare @DateStartContract date, @DateFinishContract date

Select

@DateStartContract=DateStart,

@DateFinishContract=DateFinish

from inserted

if (@DateFinishContract >= @DateStartContract)

Begin

print 'Необходимо правильно указать дату!'

ROLLBACK TRANSACTION

End

1. checkInsertBriefing

Триггер создан как ограничение таблицы Briefing на вставку и обновление полей с условием: дата начала партии не может больше даты окончания партии.

Листинг:

create trigger checkInsertBriefing on Briefing

for insert, update

as

declare @DateStartContract date, @DateFinishContract date

Select

@DateStartContract=dateStart,

@DateFinishContract=dateFinish

from inserted

if (@DateFinishContract >= @DateStartContract)

Begin

print 'Необходимо правильно указать дату!'

ROLLBACK TRANSACTION

End

# Заключение

В данной курсовой работе был проведен детальный анализ предметной области спортивный клуб. Раскрыто само понятие БД и способы его построения в соответствии с основными этапами проектирования, осуществляемыми в рамках современных информационных технологий.

В конечном итоге, на основе MSSQL была разработана база данных для спортивного клуба.

Разрабатываемая БД поможет отслеживать информацию о каждом игроке и тренере.

Использование БД в реальной автошколе позволит автоматизировать деятельность сотрудников. Создаст благоприятные условия труда, позволит всегда отражать актуальную информацию.

Поставленная цель курсовой работы достигнута, задачи решены в полном объеме.

# Список используемой литературы

1. <https://www.beboss.ru/bplans-driving>
2. <https://ru.wikipedia.org/wiki/Автошкола>
3. <https://moluch.ru/archive/218/52262/>
4. <https://itnan.ru/post.php?c=1&p=255361>