СОДЕРЖАНИЕ

[Введение 1](#_Toc165989946)

[1. Аналитическая часть 3](#_Toc165989947)

[1.1. Постановка задачи 3](#_Toc165989948)

[1.2. Характеристика предметной области 4](#_Toc165989949)

[2. Проектирование базы данных 5](#_Toc165989950)

[2.1. Определение сущностей, атрибутов, связей 5](#_Toc165989951)

[2.2. Построение инфологической модели 9](#_Toc165989952)

[2.3. Преобразование ER-модели в реляционную 10](#_Toc165989953)

[2.4. Нормализация таблиц 15](#_Toc165989954)

[3. Практическая реализация 17](#_Toc165989955)

[Заключение 42](#_Toc165989956)

# **Введение**

В последнее время всё активнее развивается информационная отрасль, связанная с созданием и использованием технических средств, методов и технологий для производства новых знаний. Эта отрасль тесно связана с развитием компьютерных технологий.

Изменяется весь уклад жизни, система ценностей человека, а именно, возрастает спрос на знания, от человека требуется способность к интеллектуальному труду и творчеству.

В связи с этим возникло противоречие между ограниченными возможностями человека по восприятию и переработке информации. Появилось большое количество избыточной информации, в которой иногда трудно ориентироваться и выбирать нужные сведения. Для решения подобных проблем применяются автоматизированные базы данных. Они стали неотъемлемой частью практически всех компьютерных систем – от отрасли до отдельного предприятия.

Реляционная СУБД – СУБД, управляющая реляционная базами данных. Понятие «реляционный» связано с разработками известного английского специалиста в области систем баз данных Эдгарда Кодда.

Эти модели характеризуются простотой структуры данных, удобным для пользователя табличными представлением и возможностью использования формального аппарата алгебры отношений и реляционного исчисления для обработки данных.

**Актуальность** выбранной мной темы обусловлена тем, что растущий объем информации сложно воспринимать и выбирать из всего массива информации необходимые данные. Для решения этой проблемы и применяются автоматизированные базы данных.

**Целью** курсовой работы: разработка базы данных для общеобразовательного учреждения «Школа».

**Задачи**:

1. Сформулировать цель разрабатываемой базы данных.
2. Дать характеристику предметной области.
3. Определить сущности, атрибуты и связи разрабатываемой базы данных.
4. Построить инфологическую модель базы данных (ER-модель).
5. Преобразовать инфологическую модель в реляционную модель базы данных.
6. Провести нормализацию таблиц.
7. Создать автоматизированную базу данных.

Структура работы включает традиционно две главы, а также введение и заключение. В первой приводится аналитический материал, описывающий рассматриваемую предметную область.

Вторая глава посвящена непосредственно проектированию автоматизированной базы данных.

# **Аналитическая часть**

## **Постановка задачи**

Исследуемой предметной областью является общеобразовательная школа. Школа – это образовательное учреждение для детей и подростков. Следовательно, школа работает с весьма большим объёмом обрабатываемой информации об учащихся, об учебных потоках (классы по году обучения), об учебных планах для каждого потока, о факультативах, доступных учащимся. Учителям необходимо постоянно следить за данными каждого учащегося. А руководству и бухгалтерии необходимо быть в курсе событий о сотрудниках школы, а также об изучаемых занятиях в школе и факультативах. Для этого нужна общая база данных, включающая всю необходимую информацию. Программа является очень актуальной на сегодняшний день, она автоматизирует работу с базой данных и предоставляет пользователю понятный и удобный интерфейс.

## **Характеристика предметной области**

Исследуемой предметной областью является общеобразовательная школа. Школа – это образовательное учреждение для детей и подростков.

Все классы в школе делятся на учебные потоки по году обучения: 1, 2, 3, …, 11. В каждом учебном потоке есть, как минимум один класс.

В каждом классе есть учащиеся, информация о которых также хранится в базе данных.

Известен учебный план для каждого потока: какие предметы и в каком объеме изучаются (количество часов в год, отведенных на изучение предмета).

Кроме изучения обязательных предметов изучаются факультативные занятия. Факультатив идентифицируется названием предмета и номером учебного потока, на который он ориентирован. Все факультативные занятия – платные. Цены на разные факультативы – разные. Цена на факультатив устанавливается за месяц посещения факультативного занятия. К посещению факультативных занятий допускаются только учащиеся, предварительно оплатившие их. Кроме того, учащиеся должны предварительно записаться на факультатив.

# **Проектирование базы данных**

## **Определение сущностей, атрибутов, связей**

Средством моделирования предметной области на этапе концептуального проектирования является модель «сущность-связь». Часто её называют ER-моделью (Entity – сущность, Relation – связь). В ней моделирование структуры данных предметной области базируется на использовании графических средств – ER-диаграмм (диаграмм «сущность-связь»). В наглядном виде они представляют связи между сущностями.

Основные понятия ER-диаграммы: *сущность*, *атрибут* и *связь*.

*Сущность* – это некоторый объект реального мира, который может существовать независимо. Сущность имеет экземпляры, отличающиеся друг от друга значениями атрибутов и допускающие однозначную идентификацию. *Атрибут* – это свойство сущности. Атрибут, который уникальным образом идентифицирует экземпляры сущности, называется *ключом*. Может быть составной ключ, представляющий комбинацию нескольких атрибутов.

Проектируется база данных, предназначенная для хранения информации о деятельности некоторой общеобразовательной школы. В этой школе все классы делятся на потоки по году обучения. Само собой, в школе есть учащиеся. Существует учебный план для каждого учебного потока, в котором указывается, какие предметы и в каком объеме изучаются на каждом потоке. Кроме обязательных предметов в школе изучаются факультативные занятия. Все факультативные занятия являются платными. Описываемую предметную область назовем ШКОЛА. В ней можно выделить четыре *сущности*: филиал, менеджер, счет, клиент.

На ER-диаграмме сущность изображается прямоугольником, в котором указывается имя сущности. Пример изображения сущности на ER-диаграмме изображен на рисунке 2.1.1.

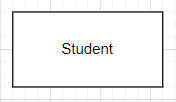


Рисунок 2.1.1. **Пример изображения сущности на ER-диаграмме**

В реальном мире существуют связи между сущностями. *Связь* представляет собой взаимодействие между сущностями. Сущность характеризуется *мощностью*, которая показывает, сколько сущностей участвуют в связи.

В рассматриваемой предметной области ШКОЛА можно выделить четыре связи:

* STREAM (УЧЕБНЫЙ ПОТОК) – ИЗУЧАЕТ – SUBJECT (ПРЕДМЕТ);
* STREAM – ИМЕЕТ – STUDENT (УЧАЩИЙСЯ);
* STUDENT – ЗАПИСЫВАЕТСЯ – ELECTIVE (ФАКУЛЬТАТИВНОЕ ЗАНЯТИЕ);
* STUDENT – ОПЛАЧИВАЕТ – ELECTIVE.

На ER-диаграмме связь изображается ромбом. Пример изображения связи на ER-диаграмме изображен на рис. 2.1.2.



Рисунок 2.1.2. **Пример изображения связи на ER-диаграмме**

Важной характеристикой связи является *тип связи*. Рассмотрим три типа связи на примере неопределенных сущностей A и B.

1. Тип «один-к-одному» (1:1): каждый экземпляр сущности A может быть связан не более чем с одним экземпляром сущности B. На рис. 2.1.3. представлена ER-диаграмма для связи типа 1:1.

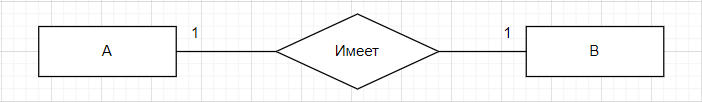


Рисунок 2.1.3. **ER-диаграмма связи 1:1**

1. Тип «один-ко-многим» (1:M): каждый экземпляр сущности A может быть связан более чем с одним экземпляром сущности B, а каждый экземпляр сущности B может быть связан не более чем с одним экземпляром сущности A. На рис. 2.1.4. представлена ER-диаграмма для связи типа 1:M.



Рисунок 2.1.4. **ER-диаграмма связи 1:M**

1. Тип «многие-ко-многим» (M:N): каждый экземпляр сущности A может быть связан с несколькими экземплярами сущности B и каждый экземпляр сущности B может быть связан с несколькими экземплярами сущности A. На рис. 2.1.5. представлена ER-диаграмма для связи типа M:N.



Рисунок 2.1.5. **ER-диаграмма связи M:N**

Каждая из сущностей рассматриваемой предметной области ШКОЛА может быть описана своим *набором атрибутов* (Таблица 1):

Таблица 1

**Наборы атрибутов сущностей предметной области ШКОЛА**

|  |  |
| --- | --- |
| Название атрибута сущности | Описание атрибута сущности |

Сущность STREAM (УЧЕБНЫЙ ПОТОК)

|  |  |
| --- | --- |
| **IdStream** | Номер учебного потока |
| Date | Дата поступления |
| Qty | Количество классов в потоке |

Сущность SUBJECT (УЧЕБНЫЙ ПРЕДМЕТ)

|  |  |
| --- | --- |
| **IdSubject** | Номер учебного предмета |
| SubjectName | Наименование предмета |
| IdStream | Номер учебного потока, на котором изучается предмет |
| SubjectHours | Объем изучения предмета |

Сущность ELECTIVE (ФАКУЛЬТАТИВНОЕ ЗАНЯТИЕ)

|  |  |
| --- | --- |
| **IdElective** | Номер факультативного занятия |
| ElectiveName | Наименование факультатива |
| ElectiveHours | Объем изучения факультатива |
| ElectivePrice | Стоимость факультатива |
| StartDate | Дата начала занятий |

Сущность STUDENT (УЧАЩИЙСЯ)

|  |  |
| --- | --- |
| **IdStudent** | Номер учащегося |
| FName | Имя учащегося |
| LName | Фамилия учащегося |
| IdStream | Номер учебного потока учащегося |
| ClassName | Наименование уч. класса учащегося |
| Address | Адрес проживания учащегося |
| Phone | Номер телефона учащегося |

**Примечание.** Ключевые атрибуты выделены жирным шрифтом.

## **Построение инфологической модели**

ER-модель рассматриваемой предметной области ШКОЛА будет иметь вид, представленный на рис. 2.2.1.

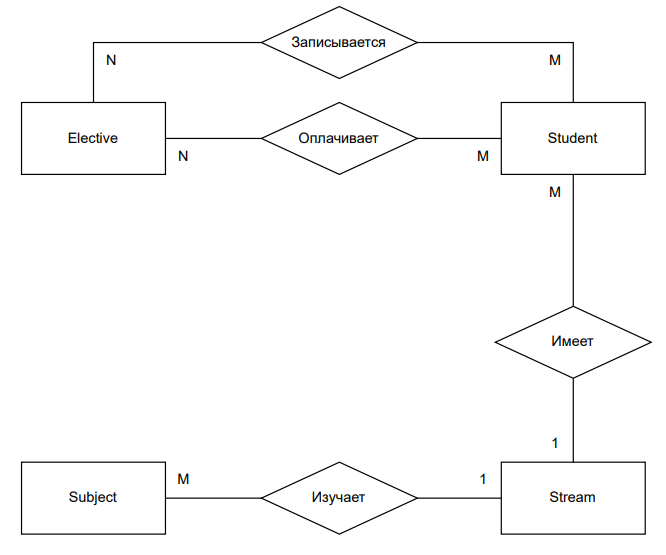


Рисунок 2.2.1. **ER-модель предметной области ШКОЛА**

## **Преобразование ER-модели в реляционную**

Концептуальные модели позволяют более точно представить предметную область, чем реляционные и другие модели. Но в настоящее время существует немного систем управления базами данных, поддерживающих эти модели. На практике наиболее распространены системы, реализующие реляционную модель. Поэтому необходим метод перевода концептуальной модели в реляционную. Такой метод основывается на формировании набора предварительных таблиц из ER-диаграмм.

Для каждой сущности создается таблица. Причем каждому атрибуту сущности соответствует столбец таблицы.

Правила генерации таблиц из ER-диаграмм опираются на два фактора – *тип связи* и *класс принадлежности сущности*. Далее изложим эти правила.

**Правило №1:**

Если связь типа 1:1 и класс принадлежности обеих сущностей является обязательным, то необходима только одна таблица. Первичным ключом этой таблицы может быть первичный ключ любой из двух сущностей.

**Правило №2:**

Если связь типа 1:1 и класс принадлежности одной сущности является обязательным, а другой – необязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности, для которой класс принадлежности является необязательным, добавляется как атрибут в таблицу для сущности с обязательным классом принадлежности.

Сущность с необязательным классом принадлежности именуется родительской, а с обязательным – дочерней. Первичный ключ родительской сущности, помещаемый в таблицу, представляющий дочернюю сущность, называется внешним ключом родительской сущности. Связь между указанными таблицами осуществляется путем связи первичного и внешнего ключа.

**Правило №3:**

Если связь типа 1:1 и класс принадлежности обеих сущностей является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь ключи обеих сущностей.

Итак, для связи типа 1:1 существует три отдельных правила формирования предварительных таблиц из ER-диаграмм.

Для связи типа 1:M существует только два правила. Выбор одного из них зависит от класса принадлежности сущности на стороне M. Класс принадлежности сущности на стороне 1 не влияет на выбор.

**Правило №4:**

Если связь типа 1:M и класс принадлежности сущности на стороне M является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ сущности на стороне 1 добавляется как атрибут в таблицу для сущности на стороне M.

**Правило №5:**

Если связь типа 1:M и класс принадлежности сущности на стороне M является необязательным, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь первичные ключи обеих сущностей.

**Правило №6:**

Если связь типа M:N, то необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь первичные ключи обеих сущностей.

К ER-модели рассматриваемой предметной области ШКОЛА, представленной на рис. 2.2.1., применимы правила 4 и 6.

*Связь* *STREAM-SUBJECT типа 1:M (один-ко-многим)*. Класс принадлежности сущности «Stream» является необязательным, а сущности «Subject» является обязательным.

Так как связь типа 1:M и класс принадлежности сущности на стороне M является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ на стороне 1 добавляется как атрибут для сущности на стороне M.

Поэтому, согласно правилу №4, необходимо сгенерировать две таблицы следующей структуры:

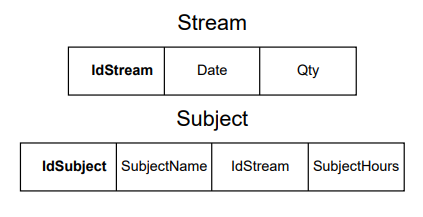


Рисунок 2.3.1. **Таблицы необходимые для связи STREAM-SUBJECT типа 1:M и их структура**

Связь между указанными таблицами будет иметь вид:

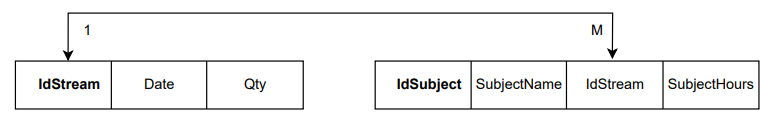


Рисунок 2.3.2. **Связь STREAM-SUBJECT типа 1:M**

*Связь STREAM-STUDENT типа 1:M (один-ко-многим)*. Класс принадлежности сущности «Stream» является необязательным, а сущности «Student» является обязательным.

Так как связь типа 1:M и класс принадлежности сущности на стороне M является обязательным, то необходимо построить таблицу для каждой сущности. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Первичный ключ на стороне 1 добавляется как атрибут для сущности на стороне M.

Поэтому, согласно правилу №4, необходимо сгенерировать две таблицы следующей структуры:

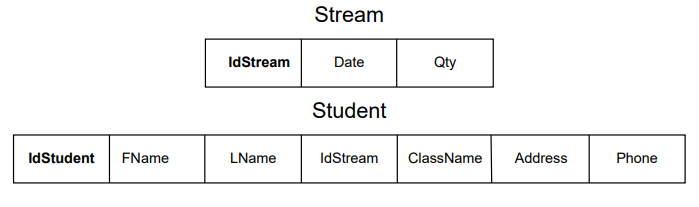


Рисунок 2.3.3. **Таблицы необходимые для связи STREAM-STUDENT типа 1:M и их структура**

Связь между указанными таблицами будет иметь вид:

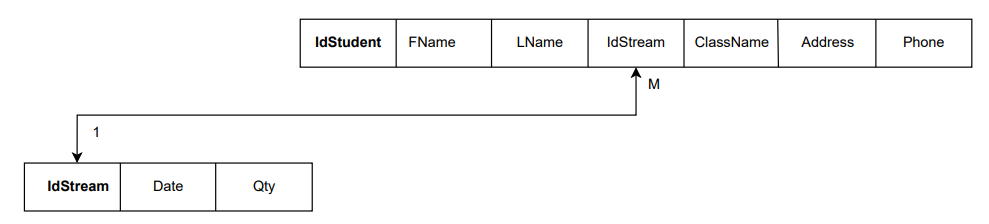


Рисунок 2.3.4. **Связь STREAM-STUDENT типа 1:M**

*Связь* *STUDENT-ELECTIVE типа M:N под названием «записывается».* Поэтому, согласно правилу №6, необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь первичные ключи обеих сущностей.

Должны быть сгенерированы три таблицы следующей структуры:

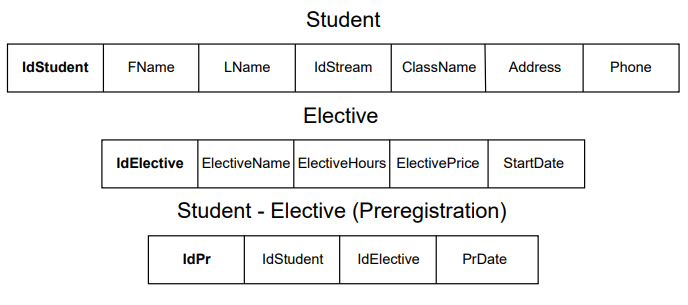


Рисунок 2.3.5. **Таблицы необходимые для связи STUDENT-ELECTIVE типа M:N под названием «записывается» и их структура**

При этом осуществляется декомпозиция связи M:N на две связи 1:M следующим образом:

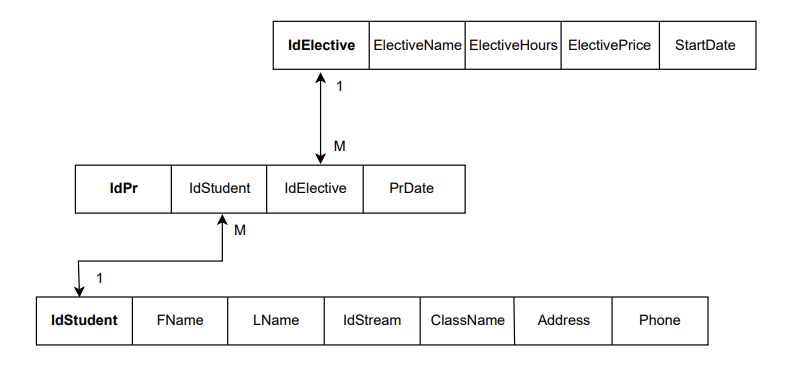


Рисунок 2.3.6. **Связь STUDENT-ELECTIVE типа M:N под названием «записывается»**

*Связь STUDENT-ELECTIVE типа M:N под названием «оплачивает».* Поэтому, согласно правилу №6, необходимо построить три таблицы – по одной для каждой сущности и одну для связи. Первичный ключ сущности должен быть первичным ключом соответствующей таблицы. Таблица для связи среди своих атрибутов должна иметь первичные ключи обеих сущностей.

Должны быть сгенерированы три таблицы следующей структуры:

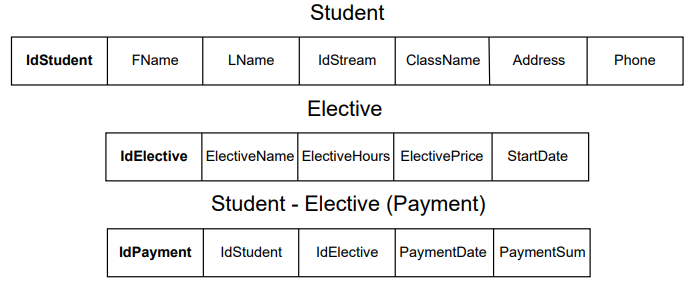


Рисунок 2.3.7. **Таблицы необходимые для связи STUDENT-ELECTIVE типа M:N под названием «оплачивает» и их структура**

При этом осуществляется декомпозиция связи M:N на две связи 1:M следующим образом:

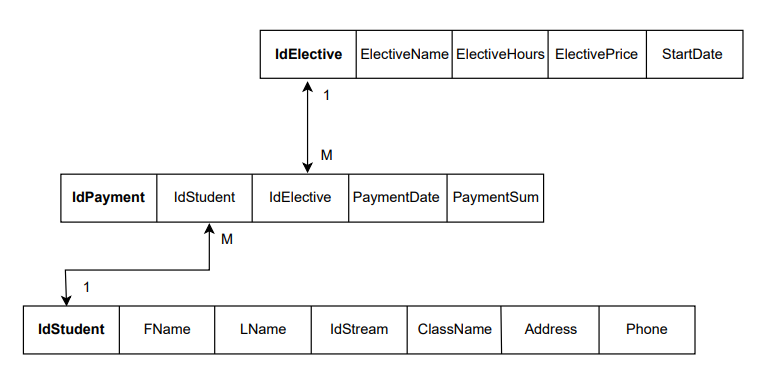


Рисунок 2.3.8. **Связь STUDENT-ELECTIVE типа M:N под названием «оплачивает»**

## **Нормализация таблиц**

Методику нормализации таблиц разработал американский ученый А.Ф. Кодд в 1970 году. Её суть сводится к приведению таблиц к той или иной форме. Были введены три нормальные формы – 1НФ, 2НФ, 3НФ. Позже стали выделять нормальную форму Бойса-Кодда (НФБК), а затем – 4НФ и 5НФ. Каждая последующая нормальная форма вводит ограничения на хранимые в базе данные.

Реляционная база данных считается *эффективной*, если все её таблицы находятся как минимум в 3НФ.

Все таблицы рассматриваемой базе данных ШКОЛА все таблицы находятся как минимум в 3НФ. На рисунке 2.4.1 представлена реляционная модель предметной области ШКОЛА после нормализации.

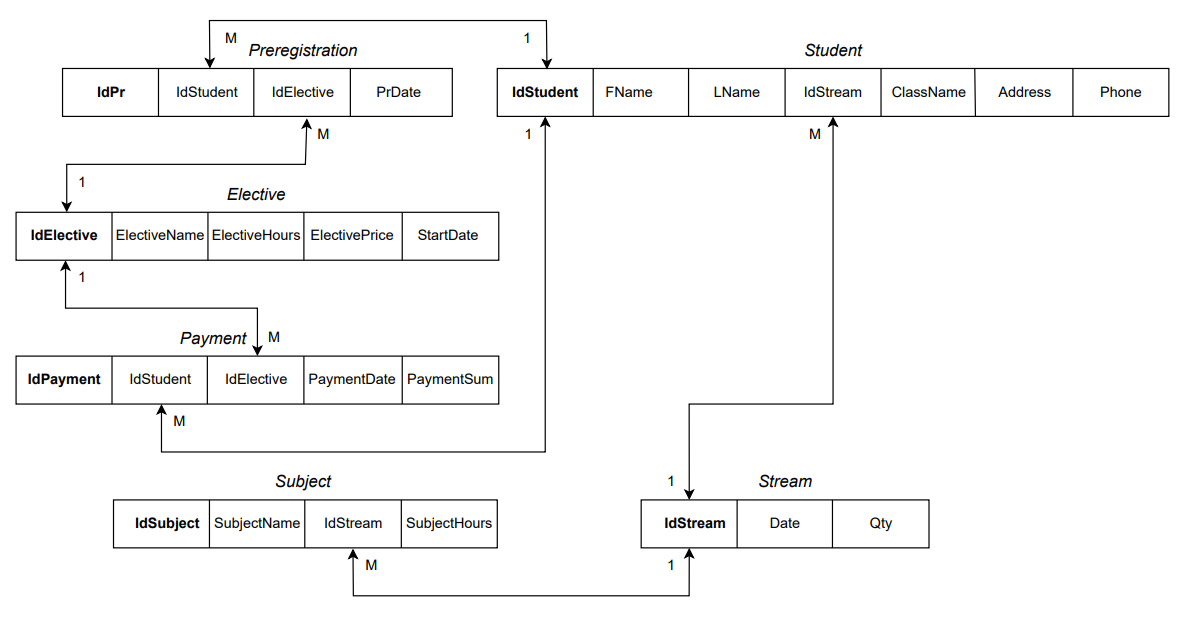


Рисунок 2.4.1. **Реляционная модель предметной области ШКОЛА после нормализации**

# **Практическая реализация**

**Создание таблиц**

Создадим в базе данных School шесть таблицы. Первая таблица, Stream, будет хранить информацию о потоках по году обучения (1-11 потоки), вторая таблица Subject – учебный план по потокам, третья, Elective, - информацию о факультативных занятиях в школе, четвертая, Student, будет содержать подробную информацию об учащихся, пятая, Preregistration, - о предварительной записи учащихся на факультативные занятия и шестая, Payment, - об оплате факультативных занятий учащимися. Ниже в таблице 1 представлены все поля этих таблиц и их основные свойства.

1. *Таблица Stream (Учебный поток)*: будет хранить информацию о потоках по году обучения. Потоки с 1 по 11.

|  |  |  |  |
| --- | --- | --- | --- |
| Имя столбца | Тип данных | Разрешить значение null | Описание |
| STREAM | | | |
| IdStream | int, identity | нет | Уникальный идентификационный номер учебного потока, на который можно ссылаться в других таблицах |
| Date | smalldatetime | нет | Дата поступления учащихся текущего учебного потока в школу |
| Qty | int | нет | Количество классов в учебном потоке |

1. *Таблица Subject (Предмет)* (учебный план): будет хранить информацию о том, какие предметы и в каком объеме изучаются на каждом учебном потоке.

|  |  |  |  |
| --- | --- | --- | --- |
| SUBJECT | | | |
| IdSubject | int, identity | нет | Уникальный идентификационный номер учебной дисциплины, на который можно ссылаться в других таблицах |
| SubjectName | nvarchar(25) | нет | Полное наименование учебного предмета |
| IdStream | int | нет | Ссылка на номер учебного потока |
| SubjectHours | int | нет | Объем изучения учебного предмета (количество часов в год) |

1. *Таблица Elective (Факультатив)*: будет храниться информация о факультативных занятиях в школе.

|  |  |  |  |
| --- | --- | --- | --- |
| ELECTIVE | | | |
| IdElective | int, identity | нет | Уникальный идентификационный номер факультативного занятия, на который можно ссылаться в других таблицах |
| ElectiveName | nvarchar(25) | нет | Полное наименование факультатива с указанием номера потока, на который он ориентирован |
| ElectiveHours | int | нет | Объем изучения факультативного занятия (количество часов в месяц) |
| ElectivePrice | money | нет | Цена за месяц занятий |
| StartDate | smalldatetime | нет | Дата начала факультативных занятий |

1. *Таблица Student (Учащийся)*: будет храниться информация обо всех учащихся.

|  |  |  |  |
| --- | --- | --- | --- |
| STUDENT | | | |
| IdStudent | int, identity | нет | Уникальный идентификационный номер учащегося, на который можно ссылаться в других таблицах |
| FName | nvarchar(20) | нет | Имя учащегося |
| LName | nvarchar(20) | нет | Фамилия учащегося |
| IdStream | int | нет | Ссылка на номер учебного потока |
| ClassName | nvarchar(20) | нет | Полное наименование класса учащегося |
| Address | nvarchar(50) | нет | Адрес проживания учащегося |
| Phone | Phone (пользовательский тип данных) | нет | Номер телефона учащегося/номер телефона одного из родителей(опекунов) учащегося |

1. *Таблица Preregistration (Предварительная запись)*: будет храниться информация о предварительной записи учащихся на факультативные занятия.

|  |  |  |  |
| --- | --- | --- | --- |
| PREREGISTRATION | | | |
| IdPr | int, identity | нет | Уникальный идентификационный номер записи учащегося на факультатив, на который можно ссылаться в других таблицах |
| IdStudent | int | нет | Ссылка на номер учащегося |
| IdElective | int | нет | Ссылка на номер факультативного занятия |
| PrDate | smalldatetime | нет | Дата записи учащегося на факультативное занятие |

1. *Таблица Payment (Оплата)*: будет храниться информация об оплате факультативных занятий учащимися.

|  |  |  |  |
| --- | --- | --- | --- |
| PAYMENT | | | |
| IdPayment | int, identity | нет | Уникальный идентификационный номер оплаты факультативного занятия, на который можно ссылаться в других таблицах |
| IdStudent | int | нет | Ссылка на номер учащегося |
| IdElective | int | нет | Ссылка на номер факультативного занятия |
| PaymentDate | smalldatetime | нет | Дата оплаты факультативного занятия учащимся |
| PaymentSum | money | нет | Сумма оплаты |

**Создание пользовательских типов данных**

SQL Server позволяет на основе системных типов данных создавать пользовательские типы со всеми предварительно заданными параметрами, включая все ограничения и умолчания. В качестве примера создадим тип данных phone, который будет использоваться в таблице STUDENT для хранения телефонного номера учащегося/одного из родителей(опекунов) учащегося. Для его создания воспользуюсь графическим интерфейсом утилиты Management Studio.

1. В дереве обозревателя объектов раскрою папки «Базы данных – School – Программирование - Типы». (база данных School уже создана). В контекстном меню узла «Определяемые пользователем типы данных» выберу команду «Создать определяемый пользователем тип данных».
2. В появившемся окне в текстовом поле «Имя» введу phone. В раскрывающемся списке «Тип данных» выберу nchar. В качестве длины введу 10. Отмечу параметр «Разрешить значения null», чтобы иметь возможность не указывать телефонный номер при добавлении нового клиента.
3. В секции «Привязки» оставлю пустые значения и щелкну на кнопке Ok. Созданный пользовательский тип данных появиться в дереве обозревателя объектов.

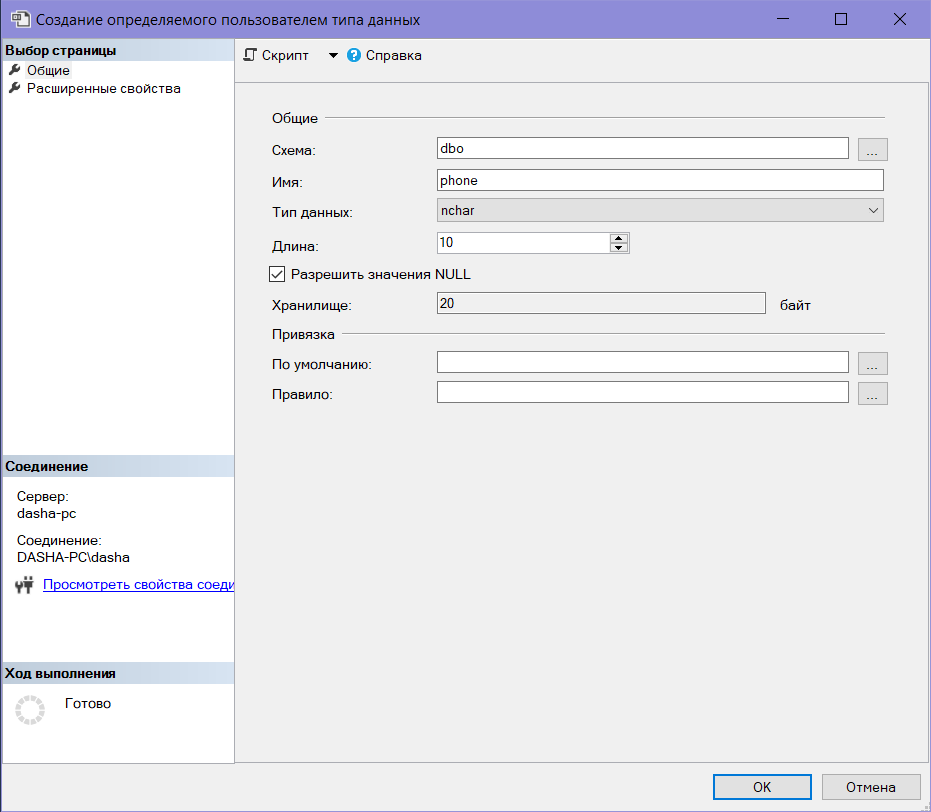


Рисунок 3.1. **Создание пользовательского типа данных phone**

Для создания таблиц воспользуюсь самым простым, графическим способом. Сначала создам таблицу Stream:

1. В дереве обозревателя объектов в базе данных School в контекстном меню узла «Таблицы» выберу команду «Создать таблицу…». В рабочей области должна появиться вкладка с конструктором таблиц.
2. В первую строку в столбце «Имя столбца» введу IdStream, в столбце «Тип данных» выберу тип int. Проверю, чтобы параметр «Разрешить значения null» был отключен.
3. В нижней половине экрана в разделе «Свойства столбцов» введу описание поля и изменю значение параметра «Спецификация идентификатора / (Идентификатор)» на «Да» для того, чтобы значения номера учебного потока формировались автоматически. Свойство «Идентифицирующий столбец» (Identity), обычно используемое совместно с типом данных int, предназначено для автоматического приращения значения на единицу при добавлении каждой новой записи.
4. Аналогичным образом введу описания всех остальных полей и закрою окно конструктора таблиц. Введу в качестве имени таблицы Stream. Вновь созданная таблица появиться в дереве обозревателя объектов в папке «Таблицы».

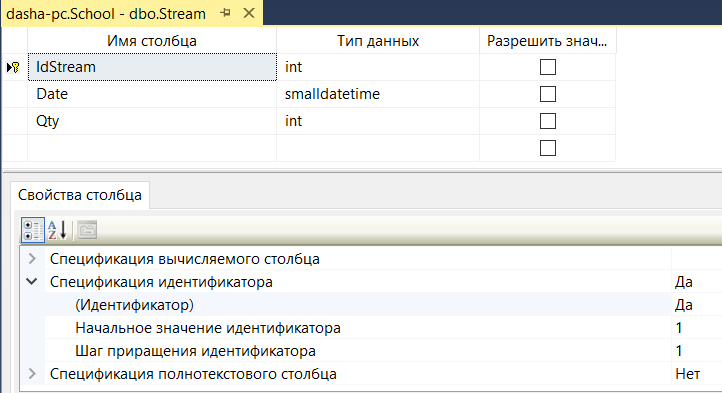


Рисунок 3.2. **Создание таблицы Stream с помощью графического способа**

На примере создания таблицы Stream через конструктор таблиц создам остальные таблицы и заполню их пробными данными.

**Созданные и заполненные пробными данными таблицы:**

1. Таблица Stream:

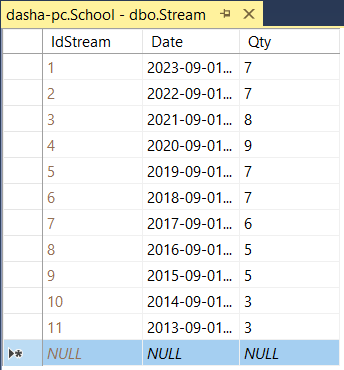


Рисунок 3.3. **Таблица Stream, заполненная пробными данными об учебных потоках школы**

1. Таблица Subject:

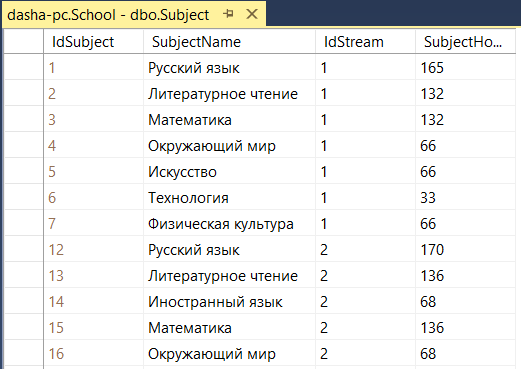


Рисунок 3.4. **Таблица Subject, заполненная пробными данными об изучаемых в школе предметах**

1. Таблица Student:



Рисунок 3.5. **Таблица Student, заполненная пробными данными об учащихся школы**

1. Таблица Elective:

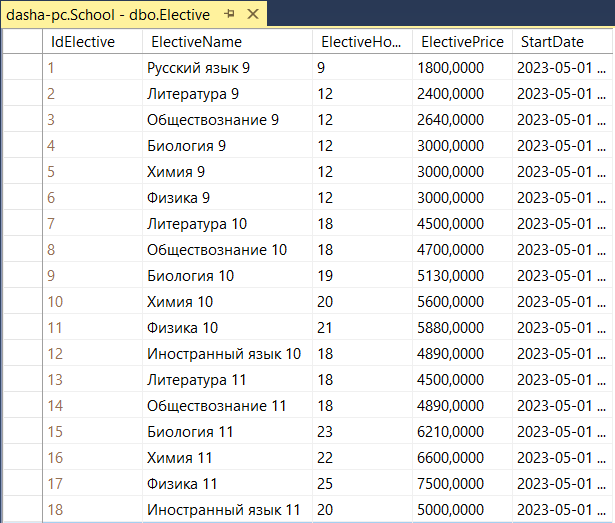


Рисунок 3.6. **Таблица Elective, заполненная пробными данными о факультативных занятиях**

1. Таблица Preregistration:

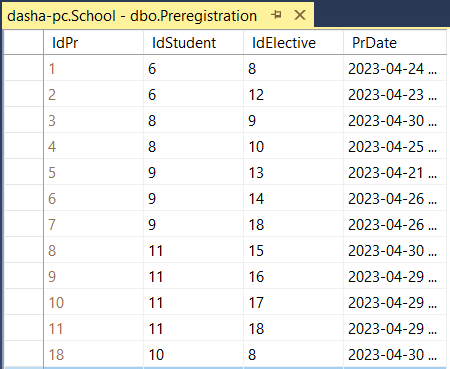


Рисунок 3.7. **Таблица Preregistration, заполненная пробными данными о записях учащихся на факультативные занятия**

1. Таблица Payment:

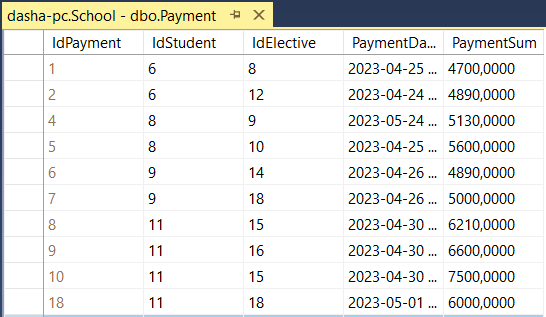


Рисунок 3.8. **Таблица Payment, заполненная пробными данными об оплате факультативных занятий учащимися**

**Создание ограничений**

Перед тем как начать работать с таблицами следует ограничить вводимые в них данные в целях обеспечения так называемой целостности данных, т. е. ограничить возникновение в базе данных некорректных или противоречивых данных вследствие добавления, изменения или удаления какой-либо записи. Существует четыре типа целостности данных: доменная, сущностная, ссылочная и пользовательская.

**Обеспечение доменной целостности.** Ограничение диапазона данных, вводимых пользователем в поле. Основными инструментами обеспечения доменной целостности являются ограничения проверки и значения по умолчанию.

**Использование проверочных ограничений**

Ограничения на проверку используются для ограничения данных, принимаемых полем, даже если они имеют корректный тип. Например, поле Qty (количество учебных классов в потоке) имеет тип int, т.е. теоретически это поле может принимать отрицательные значения. Это может стать проблемой, поскольку не может быть отрицательное количество классов в учебном потоке. Поэтому мне необходимо создать ограничение на проверку, запрещающее вводить в это поле отрицательные значения.

1. В контекстном меню папки «Ограничения» таблицы Stream выберу команду «Создать ограничение».
2. В открывшемся окне «Проверочные ограничения» заполню следующие поля, как на рисунке 3.9:
   * Имя: CK\_Qty
   * Выражение: ([Qty]>=(0)). Данное выражение описывает ограничение, принимающее значение поля Qty>=0.
   * Описание: Ограничение на значения количества классов в потоке.

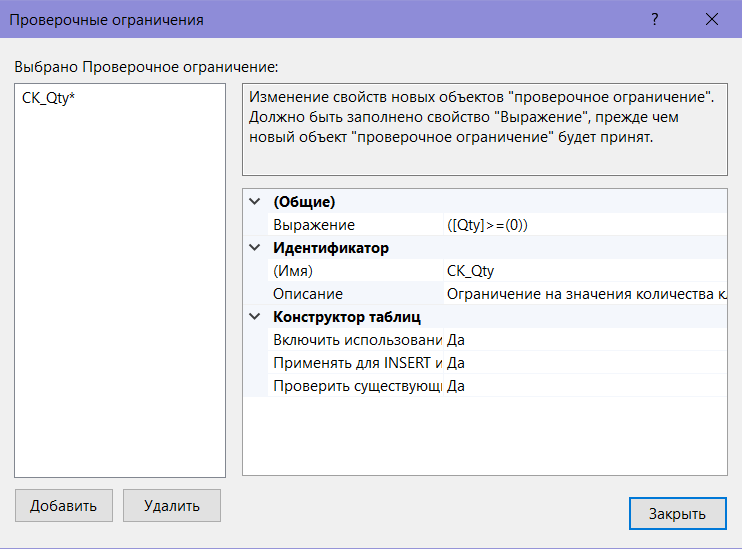


Рисунок 3.9. **Создание проверочного ограничения для поля Qty таблицы Stream**

1. Щелкну на кнопке «Закрыть» и закрою конструктор таблиц с сохранением изменений.

По примеру создания проверочного ограничения для поля Qty в таблице Stream создам следующие ограничения:

Таблица 2

**Проверочные ограничения**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица | Имя столбца | Имя ограничения | Выражение | Описание |
| Subject | SubjectHours | CK\_SubjectHours | ([SubjectHours]>(0)) | Ограничение на количество часов в год, отведенных на изучение предмета |
| Elective | ElectiveHours | CK\_ElectiveHours | ([ElectiveHours]>(0)) | Ограничение на количество часов в месяц, отведенных на изучение факультатива |
| Elective | ElectivePrice | CK\_ElectivePrice | ([ElectivePrice]>(0)) | Ограничение на стоимость факультативного занятия в месяц |
| Payment | PaymentSum | CK\_PaymentSum | ([PaymentSum]>(0)) | Ограничение на сумму оплаты факультативного занятия |

**Использование значений по умолчанию**

Установка для полей значений по умолчанию это отличный способ избавить пользователя от излишней работы, если значения этих полей во всех записях, как правило, принимают одни и те же значения. Так в таблице Stream вполне логично определить по умолчанию значение поля Date (дата поступления) в виде текущей даты. В этом случае при добавлении записи о новом учебном потоке в случае пропуска этого поля оно будет автоматически заполняться значением системной даты. Для создания такого свойства необходимо выполнить следующие шаги:

1. Раскрою папку «Столбцы» таблицы Stream и в контекстном меню поля Date выберу команду «Изменить».
2. В свойстве столбца «Значение или привязка по умолчанию» введу getdate(), как это изображено на рисунке 3.10. Эта функция T-SQL возвращает текущую системную дату.
3. Щелкну на кнопке Сохранить и выйду из конструктора таблиц.

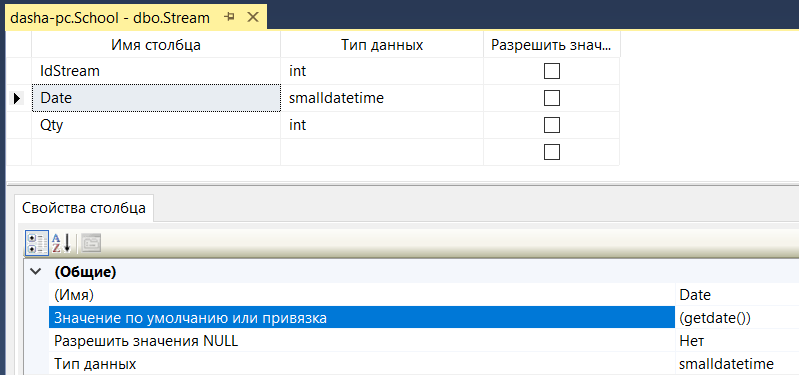


Рисунок 3.10. **Создание значения по умолчанию для поля Date таблицы Stream**

По примеру сделать создания значения по умолчанию задать следующие значения по умолчанию (таблица 2):

Таблица 3

**Значения по умолчанию**

|  |  |  |
| --- | --- | --- |
| Таблица | Имя столбца | Выражение |
| STREAM | Qty | (0) |
| ELECTIVE | StartDate | getdate() |
| PREREGISTRATION | PrDate | getdate() |
| PAYMENT | PaymentDate | getdate() |

**Обеспечение сущностной целостности**. Обеспечение гарантии уникальности записей в таблицах и предотвращение их дублирования. Основными инструментами обеспечения целостности сущностей являются первичные ключи и ограничения уникальности.

**Создание первичных ключей**

Первичный ключ используется для обеспечения гарантии уникальности каждой записи в таблице. Он состоит из одного (простой ключ) или нескольких (составной ключ) столбцов с гарантированно уникальными значениями для каждой записи таблицы. Если пользователь попытается ввести в поля первичного ключа дублирующее значение будет сгенерирована ошибка и модификация данных будет отменена.

В качестве примера создадим первичный ключ для таблицы Stream. В данном случае идеальным кандидатом на роль первичного ключа выступает столбец IdStream, поскольку значения, содержащиеся в нем, являются уникальными по определению (для него установлено свойство identity). Для создания первичного ключа в таблице Stream выполню следующие шаги:

1. В контекстном меню таблицы Stream выберу команду «Проект».
2. В окне конструктора таблиц щелкну правой кнопкой мыши на поле IdStream и выберу команду «Задать первичный ключ». Теперь слева от поля IdStream отображается значок ключа, указывающий, что поле является первичным ключом (рисунок 3.11).
3. Закрою конструктор таблиц с сохранением изменений.

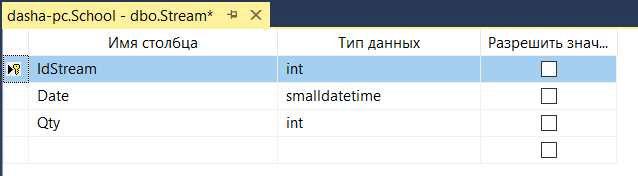


Рисунок 3.11. **Создание первичного ключа для поля IdStream таблицы Stream**

Аналогичным образом создам первичные ключи для остальных таблиц в соответствие с ниже приведенной таблицей (таблица 4).

Таблица 4

**Первичные ключи**

|  |  |
| --- | --- |
| Таблица | Первичный ключ |
| Stream | IdStream |
| Subject | IdSubject |
| Elective | IdElective |
| Student | IdStudent |
| Preregistration | IdPr |
| Payment | IdPayment |

**Использование ограничений на уникальность**

Между ограничениями первичного ключа и ограничениями на уникальность существует два отличия. Первое состоит в том, что первичные ключи используются вместе с внешними ключами для обеспечения целостности ссылок (рассматривается в следующем разделе). Второе отличие заключается в том, что ограничения на уникальность позволяют вставлять в его поля пустые значения (null), чего нельзя делать с первичными ключами. Во всем остальном они служат одной цели – обеспечить уникальность данных, вставляемых в поле. Ограничение на уникальность следует использовать в тех случаях, когда нужно гарантировать, что дублирующие значения не будут добавляться в поле, не являющееся частью первичного ключа, в частности, все потенциальные ключи должны быть организованы в виде ограничений уникальности. Хорошим примером такого поля, требующего ограничение на уникальность, является поле ИНН или серия и номер паспорта, поскольку эти поля должны быть уникальными у каждого человека. Такого идеального кандидата на роль уникального ограничения в нашей таблице Student нет. Поэтому создадим его по полю SubjectName в таблице Subject, которое повторяться у разных предметов не должно.

1. Для открытия конструктора таблиц в контекстном меню таблицы Subject выберу команду «Проект». На панели инструментов нажму на кнопку «Управление индексами и ключами».
2. В открывшемся окне «Индексы и ключи» щелкну кнопку «Добавить» и введу следующие параметры для нового уникального ключа (рисунок 3.12):
   * Столбцы: SubjectName
   * Тип: Уникальный ключ
   * (Имя): CK\_ SubjectName

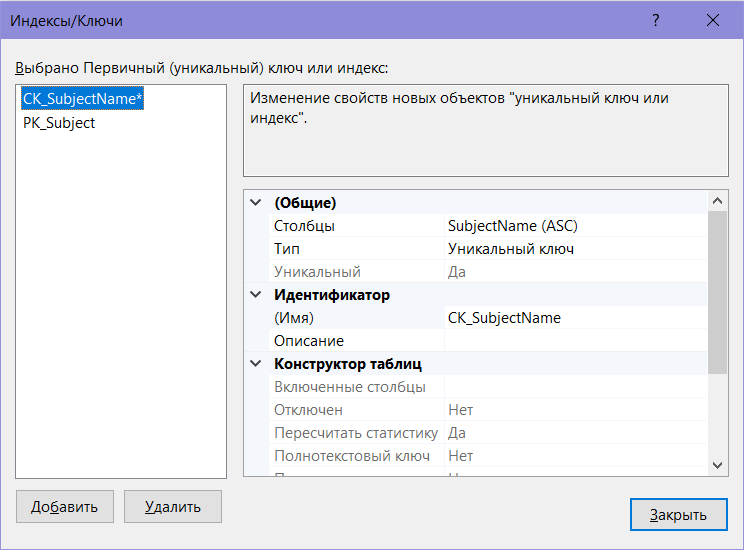


Рисунок 3.12. **использование ограничения на уникальность для поля SubjectName таблицы Subject**

1. Закрою конструктор таблиц с сохранением изменений.

По примеру создадим уникальное ограничение по полю ElectiveName в таблице Elective (рисунок 3.13), которое повторяться у разных факультативных занятий не должно.

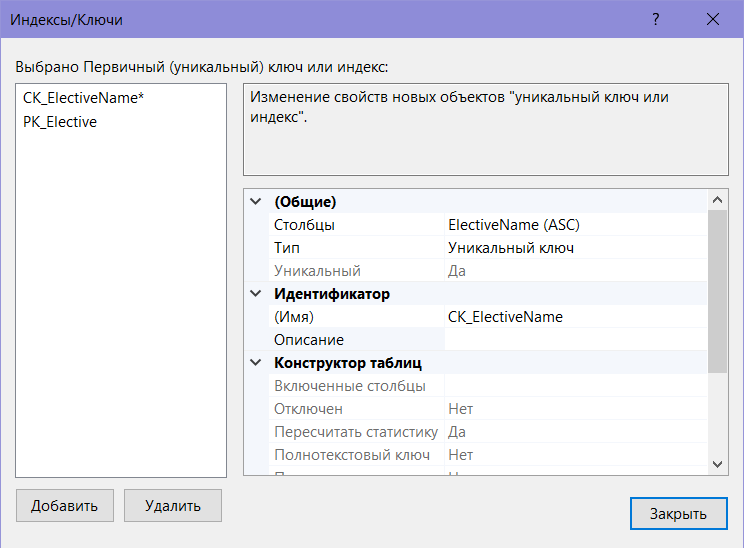


Рисунок 3.13. **Использование ограничения на уникальность для поля ElectiveName таблицы Elective**

**Обеспечение целостности ссылок**

Сейчас в базе данных School имеются шесть таблиц, которые тесно взаимосвязаны между собой и соответственно данные, содержащиеся в них, должны быть согласованы и непротиворечивы. Например, в таблице Subject не должно быть записей о предметах для учебного потока, данные о котором отсутствуют в таблице Stream. Чтобы гарантировать отсутствия в базе данных таких записей необходимо обеспечить целостность ссылок.

Суть обеспечения целостности ссылок очевидна из названия: данные в одной таблице, ссылающиеся на данные из другой таблицы, защищены от некорректного обновления. В терминологии SQL Server это называется *декларативной ссылочной целостностью* и достигается путем связывания первичного ключа одной из таблиц с внешним ключом другой таблицы (создается так называемое ограничение внешнего ключа).

*Внешний ключ* используется в комбинации с первичным для связывания двух таблиц по общему столбцу (столбцам). К примеру, можно связать таблицы Stream и Subject по столбцу IdStream, который присутствует в обеих таблицах. Поскольку поле IdStream таблицы Stream является его первичным ключом можно использовать поле IdStream таблицы Subject в качестве внешнего ключа, который свяжет эти две таблицы. После организации такого ограничения будет невозможно добавить запись в таблицу Subject, если в таблице Stream нет записи с соответствующим значением IdStream. Кроме того, при отсутствии каскадирования невозможно удалить запись из таблицы Stream при наличии связанных с ней записей в таблице Subject, поскольку нельзя оставлять предмет без информации об учебном потоке, на котором он изучается. Для создания описанного ограничения внешнего ключа в Management Studio выполню следующие шаги:

1. В контекстном меню папки «Ключи» таблицы Subject выберу команду «Создать внешний ключ…».

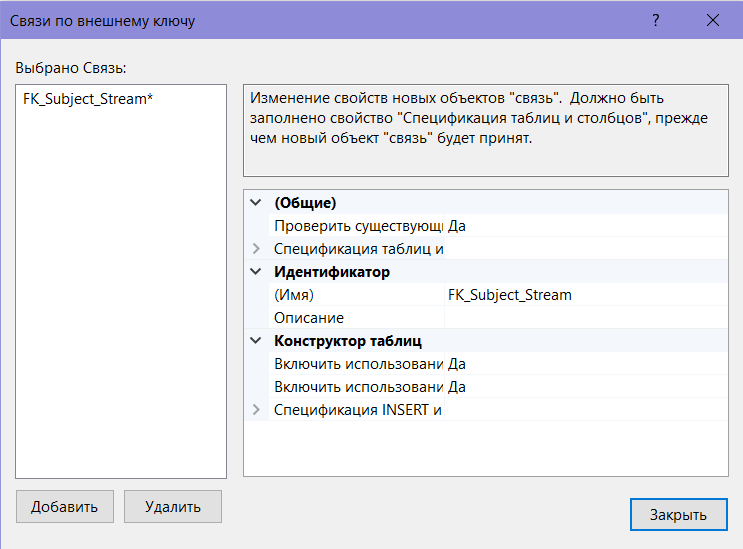


Рисунок 3.14. **Создание внешнего ключа для поля IdStream таблицы Subject**

1. В открывшемся окне «Связи по внешнему ключу» заполните следующие поля (рисунок 3.14):
   * (Имя): FK\_Subject\_Stream
   * Спецификация таблиц и столбцов: для заполнения данного блока щелкните на кнопке с многоточием и в появившемся окне «Таблицы и столбцы» в качестве таблицы первичного ключа выберите Stream, а полей связи – IdStream (рисунок 3.15).

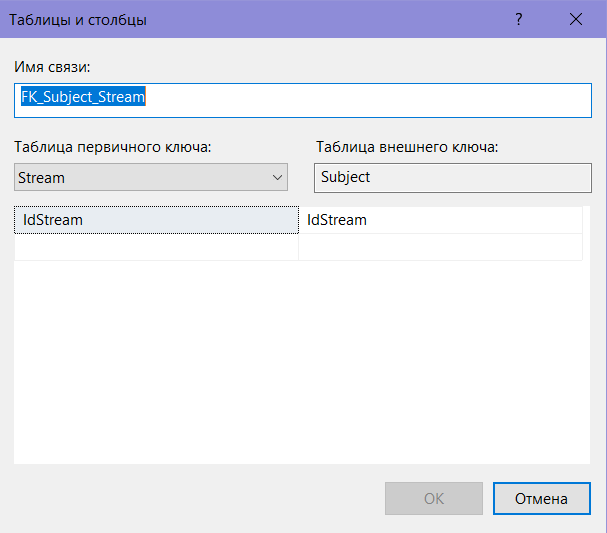


Рисунок 3.15. **Заполнение полей в окне «Таблицы и столбцы»**

1. Закрою все открывшиеся окна с сохранением изменений.

По вышеприведенному примеру создам внешние ключи в соответствие с таблицей 5:

Таблица 5

**Внешние ключи**

|  |  |  |  |
| --- | --- | --- | --- |
| Главная таблица | Подчиненная таблица | Поле связи (внешний ключ) | Правила каскадирования |
| Stream | Subject | IdStream | Без действия |
| Stream | Student | IdStream | Без действия |
| Student | Preregistration | IdStudent | Каскадно |
| Student | Payment | IdStudent | Каскадно |
| Elective | Preregistration | IdElective | Каскадно |
| Elective | Payment | IdElective | Каскадно |

**Использование диаграмм баз данных**

*Диаграммы базы данных* представляют собой графическое отображение схемы (целиком или частично) базы данных с таблицами и столбцами, а также связей между ними. Создадим диаграмму базы данных:

1. В контекстном меню папки «Диаграммы базы данных» выберу команду «Создать диаграмму базы данных».
2. В диалоговом окне «Добавление таблиц» выберу все таблицы и нажму на кнопку «Добавить».
3. Добавив таблицы, щелкну на кнопке «Закрыть» и увижу созданную диаграмму базы данных (на рисунке 3.16 представлен окончательный вид диаграммы).

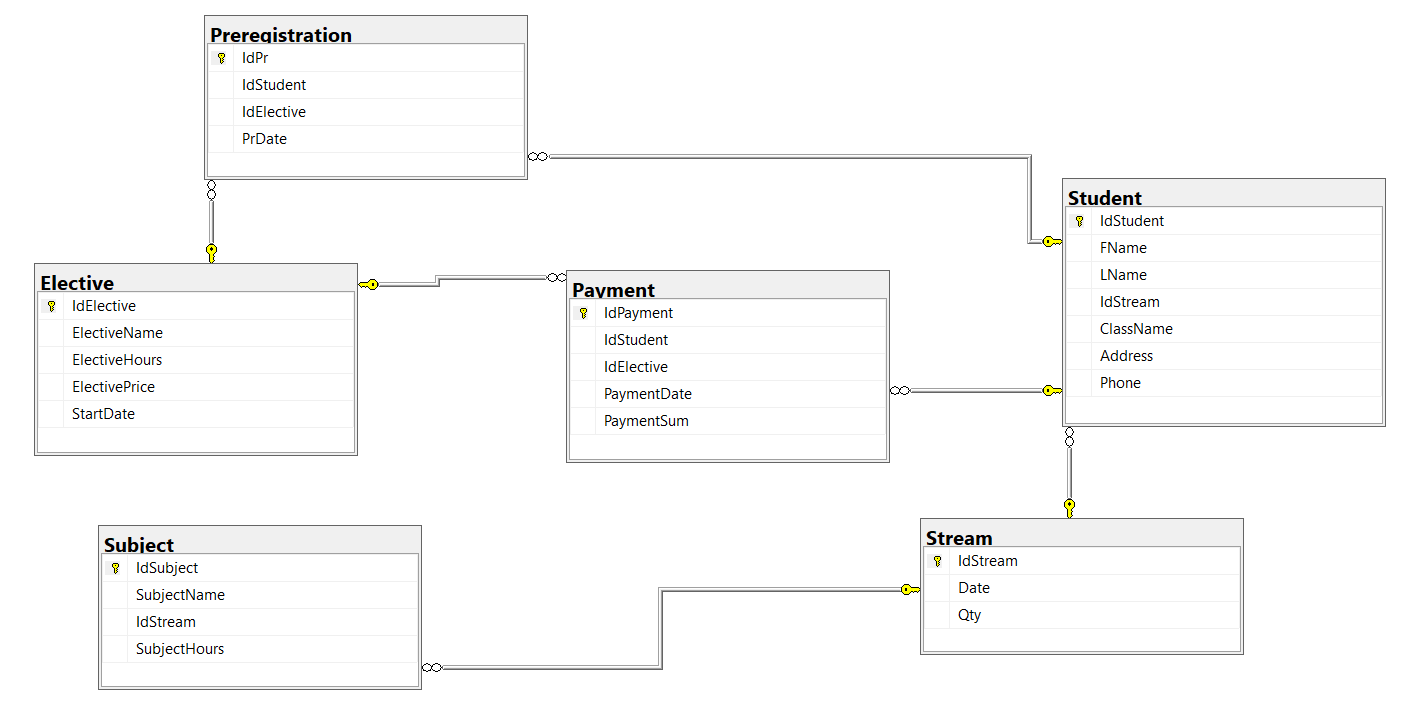


Рисунок 3.16. **Диаграмма базы данных School**

**Проектирование хранимых процедур**

При разработке приложений, основанных на платформе «клиент-сервер», для облегчения выполнения каких-либо операций с данными используются механизмы, при помощи которых можно создавать подпрограммы, работающие на сервере и управляющие процессами обработки информации. Эти механизмы носят название хранимых процедур.

В курсовой работе была разработана хранимая процедура «getCount», реализующая возврат количества учащихся, записавшихся на тот или иной факультатив (по IdElective, который задаем при выполнении процедуры): проверяем оплатил ли учащийся это факультатив (проверяем записался ли учащийся заранее на этот факультатив); и находим сумму COUNT(\*) учащихся, которые зачислены на этот факультатив.

Код:

CREATE PROCEDURE [dbo].[spr\_getCount]

@IdElective int

AS

BEGIN

SELECT DISTINCT Elective.IdElective, Elective.ElectiveName, COUNT(Payment.IdStudent) AS StudQty

FROM Elective, Payment

WHERE Payment.IdElective=Elective.IdElective AND Elective.IdElective=@IdElective

GROUP BY Elective.IdElective, Elective.ElectiveName

END

Код на исполнение процедуры:

EXEC spr\_getCount

@IdElective=9

Результат работы хранимой процедуры изображена на рисунке 3.17:

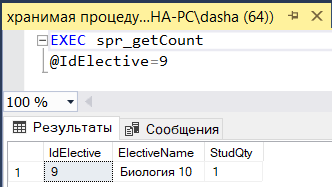


Рисунок 3.17. **Результат выполнения хранимой процедуры getCount**

**Проектирование триггеров**

Триггеры являются особой разновидностью хранимых процедур, выполняемых автоматически (срабатывающих) при изменении каких-либо данных таблицы. Особенно полезным свойством триггеров явлеятся то, что они имеют доступ к образам записи до и после модификации, так, можно сравнить две записи и принять соответствующее решение.

В курсовой работе мной было разработано три триггера:

1. Триггер «Preregistration»: вывод сообщения 'Для оплаты факультативного занятия нужно заранее записаться на него!', если студент захочет оплатить факультатив, на который он не записан заранее (в таблице Preregistration).

Код:

CREATE TRIGGER [dbo].[tr\_Preregistration] ON [dbo].[Payment]

AFTER INSERT,UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF NOT EXISTS(SELECT \*

FROM inserted AS i INNER JOIN

Preregistration AS pre ON i.IdStudent = pre.IdStudent

WHERE i.IdElective=pre.IdElective)

BEGIN

ROLLBACK TRANSACTION

RAISERROR('Для оплаты факультативного занятия нужно заранее записаться на него!',16,1)

END

END

1. Триггер «PrDate»: вывод сообщения 'Запись на факультативное занятие может осуществляться только до начала занятий! ', если учащийся захочет записаться на факультатив после или в день даты начала факультатива (указана в таблице Elective).

Код:

CREATE TRIGGER [dbo].[tr\_PrDate] ON [dbo].[Preregistration]

AFTER INSERT,UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS(SELECT \*

FROM inserted AS i INNER JOIN

Elective AS el ON i.IdElective=el.IdElective

WHERE i.PrDate>(el.StartDate-1)

)

BEGIN

ROLLBACK TRANSACTION

RAISERROR('Запись на факультативное занятие может осуществляться только до начала занятий!',16,1)

END

END

1. Триггер «PayDate»: вывод сообщения 'Оплата факультативного занятия может осуществляться только до начала занятий или в первый день начала факультативных занятий!', если студент захочет оплатить факультатив после даты начала факультатива (указана в таблице Elective).

Код:

CREATE TRIGGER [dbo].[tr\_PayDate] ON [dbo].[Payment]

AFTER INSERT,UPDATE

AS

BEGIN

SET NOCOUNT ON;

IF EXISTS(SELECT \*

FROM inserted AS i INNER JOIN

Elective AS el ON i.IdElective=el.IdElective

WHERE i.PaymentDate>el.StartDate

)

BEGIN

ROLLBACK TRANSACTION

RAISERROR('Оплата факультативного занятия может осуществляться только до начала занятий или в первый день начала факультативных занятий!',16,1)

END

END

# **Заключение**

Разработанная в ходе выполнения курсовой работы база данных «Школа», является актуальной на сегодняшний день и имеет большую практическую значимость. Она помогает работе сотрудников школы по сбору и обработке данных.

В результате выполнения данной курсовой работы были решены задачи, поставленные в начале работы. Была разработана структура базы данных; в программу было включено выполнение хранимой процедуры и использование триггеров. При этом были учтены все требования, выдвинутые в начале выполнения данной работы.

Разработанная автоматизированная база устойчиво выполняет все свои функции, и теперь стоит задача сделать её более совершенной и более расширенной.