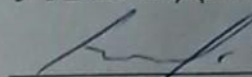




МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ  
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ  
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ  
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ  
ИМЕНИ И.С. ТУРГЕНЕВА»

Кафедра информационных систем

УТВЕРЖДАЮ:

 Зав. кафедрой

« 6 » марта 2018 г.

**ЗАДАНИЕ**  
на курсовой проект

по дисциплине «Базы данных»

Студент      Пылаева Д.А.

Шифр 160660

Институт приборостроения, автоматизации и информационных технологий

Направление подготовки 09.03.04 «Программная инженерия»

Направленность (профиль) Промышленная разработка программного обеспечения

Группа 61-ПГ

1 Тема курсовой работы

«Разработка базы данных в соответствии с заданием»

2 Срок сдачи студентом законченной работы « 6 » июня 2018



### 3 Исходные данные

Предметной областью курсового проекта является учет детей в детском дошкольном образовательном учреждении (детском саду)

### 4 Содержание курсовой работы

Анализ предметной области

Проектирование структуры базы данных

Реализация на языке SQL

### 5 Отчетный материал курсовой работы

Пояснительная записка курсовой работы; SQL-скрипты для создания и работы с базой данных

Руководитель \_\_\_\_\_ Рыженков Д.В.

Задание принял к исполнению: « 5 » марта 2018

Подпись студента \_\_\_\_\_

## Содержание

Введение	4
1 Анализ предметной области	5
2 Проектирование базы данных	7
2.1 Разработка концептуальной схемы базы данных	7
2.2 Описание сущностей и определение ключевых полей	8
2.3 Нормализация таблиц	12
2.4 Разработка физической схемы базы данных	18
3 Реализация БД	22
3.1 Построение запросов к базе данных с помощью языка SQL	23
3.2 Разработка хранимых процедур и триггеров	24
Заключение	26
Список используемой литературы	27
Приложение А	28
Приложение Б	38
Приложение В	39

## **Введение**

В настоящее время люди привыкли работать с большими объемами данных, но когда эти данные находятся разрозненно и бесструктурно, найти нужную информацию достаточно трудно. Чтобы ускорить время поиска и получения необходимой информации и используются различные базы данных.

Объект исследований – реляционная модель данных в описании деятельности предприятия.

Предмет исследований – изучение технологии проектирования и разработки базы данных на основе реляционной модели.

Целью данной работы является закрепление знаний по предмету «базы данных», а так же применение их на практике.

Задачи данной курсовой работы – изучение теоретического материала в рамках данной дисциплины, проектирование собственной базы данных по заданной теме, разработка её объектов и реализация на языке SQL.

## 1 Анализ предметной области

Данная база данных проектируется для обычного среднестатистического муниципального бюджетного дошкольного общеобразовательного учреждения.

Существует детский сад, который может принимать на обучения детей с 1 года 6 месяцев до 7 лет. Он имеет свою структуру и в его состав входят четырнадцать групп: две группы для детей раннего возраста, 5 общеразвивающих групп и 7 групп компенсирующей направленности (для детей с нарушением зрения или нарушением речи). Каждая группа имеет название, прикрепленных к ней воспитателей и помощника воспитателей, а так же максимальное количество детей. В офтальмологических и логопедических группах детей меньше, чем в остальных.

Воспитатели ведут журнал посещения детей, в котором отмечается, в какие дни ребенок посещал детский сад.

При поступлении ребенка в ДС следует приложить следующие сведения: свидетельство о рождении, паспорт и номер телефона одного из родителей.

С родителями заключается договор, в котором указываются данные родителя (или опекуна); ФИО и дата рождения ребенка и его место проживания; стоимость дополнительных общеобразовательных услуг (наименование, перечень и форма предоставления которых определены в приложении к договору); дата подписания договора и подпись заказчика.

На каждого ребенка заводится медицинская карта, в которой хранятся данные о заболеваниях, прививках и аллергических реакциях. Начальные данные об этом должны предоставить родители.

Ребенка могут перевести в другую группу, если у него обнаружатся проблемы со здоровьем. При переводе детей из группы в группу или из

детского сада в детский сад, заведующая составляет приказ, хранящийся в учреждении.

## 2 Проектирование базы данных

### 2.1 Разработка концептуальной схемы базы данных

Перед тем, как начать создавать базу данных, необходимо представить предметную область в виде концептуальной схемы, позволяющей наглядно определить необходимые сущности и их атрибуты.

Концептуальная модель — это отражение предметной области, для которой разрабатывается база данных. Все объекты обозначаются в виде прямоугольника, характеристики объекта — в виде овала, а связи между объектами — ромбами.

Изучив предметную область можно сразу выделить следующие объекты: ребёнок, приказ, группа, журнал, работник детского сада, дополнительное занятие, договор, медицинская карта.

У каждого из этих объектов есть определенные характеристики.

- Характеристиками **ребёнка** является его имя, фамилия, отчество и дата рождения;
- Характеристиками **договора** являются имя, фамилия и отчество официального представителя, номер свидетельства о рождении ребенка, текст договора и дата его заключения;
- **Журнал** имеет следующие характеристики: дата отметки о посещении и сама отметка (был/не был);
- **Медицинская карта** характеризуется списком перенесенных ребенком болезней, сделанными прививкам и имеющимися у него аллергиями;
- Характеристикой **приказа** является дата его подписания;
- **Работник детского сада** характеризуется ФИО и должностью;
- Характеристиками **группы** являются название, тип и лимит;
- **Дополнительные занятия** характеризуются наименованием и ценой.

Все эти объекты тесно связаны между собой. Каждый **ребёнок** имеет медицинскую карту. Каждого **ребёнка** отмечают в журнале. При



поступлении или переводе **ребенка** составляется **приказ**. Этот приказ закрепляет **ребёнка** за определенной **группой**. Каждая **группа** находится под контролем нескольких **работников детского сада**. **Работник** (воспитатель) ведет **журнал**. **Журнал** закреплен за определенной **группой**. Работник, не являющийся воспитателем, может вести **дополнительные занятия**. Любой **ребёнок** может посещать **дополнительные занятия**. При поступлении **Ребёнка** в детский сад составляется **договор**, в которые ребенок записан.

Результат описанного выше анализа представлен на рисунке 2.1.

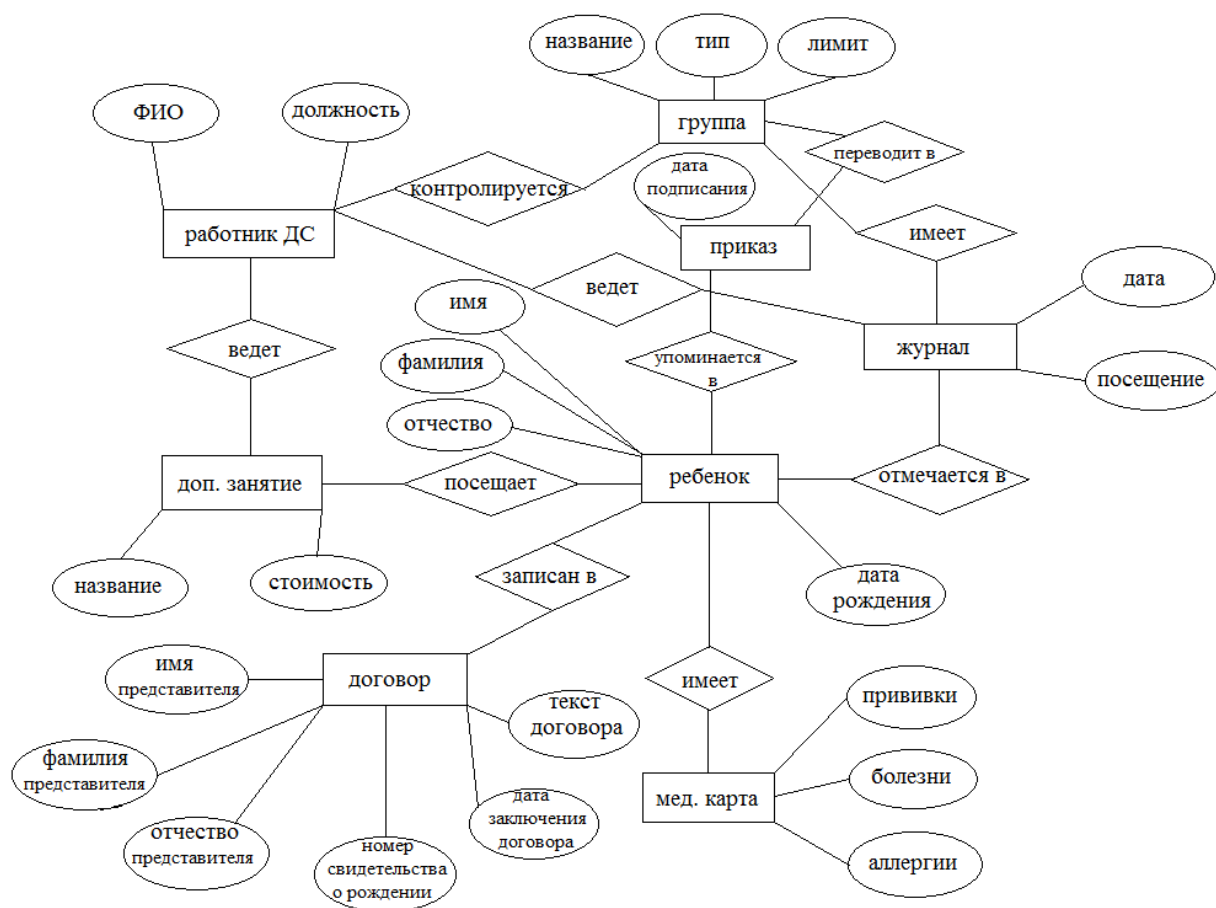


Рисунок 2.1 — Концептуальная схема базы данных

## 2.2 Описание сущностей и определение ключевых полей

Для составления логической схемы, проанализировав получившуюся концептуальную схему, мы можем выявить следующие сущности со следующими атрибутами:

Сущность «Ребенок» предназначена для хранения информации о ребенке и имеет следующие атрибуты: «Имя», «Фамилия», «Отчество», «Дата рождения».

Сущность «Группа» хранит сведения о группе и имеет атрибуты: «Название», «Тип», «Лимит».

Сущность «Приказ» хранит сведения о соответствующем приказе и имеет следующие атрибуты: «Дата подписания», «Ребёнок», «Группа».

Сущность «Работник детского сада» имеет следующие атрибуты: «ФИО», «Должность».

Сущность «Договор» имеет атрибуты: «Имя представителя», «Фамилия представителя», «Отчество представителя», «Текст договора», «Дата заключения договора», «Номер свидетельства о рождении».

Сущность «Журнал» имеет следующие атрибуты: «Дата», «Посещение», «Группа», «Ребёнок», «Ответственный».

Сущность «Дополнительное занятие» имеет атрибуты: «Наименование дисциплины», «Стоимость», «Педагог».

Также требуется сущность «Медицинская карта» состоящая из следующих атрибутов: «Болезни», «Прививки», «Аллергические реакции».

В каждой сущности должно быть определено ключевой атрибут. Этот, так называемый первичный ключ, обеспечивает уникальность записей в таблице, препятствуя вводу повторяющихся данных. С помощью ключа обеспечивается быстрый поиск требуемых данных, а также хранение и обработку.

В качестве подобного ключа разрешается использовать как одно, так и несколько полей, но с одним условием: поле или их сочетания должны быть абсолютно уникальными.

Для имеющихся у нас сущностей определены следующие первичные ключи:

- «Код ребёнка»\* для сущности «Ребёнок»;
- «Код группы»\* для сущности «Группа»;

- «Номер приказа»\* для сущности «Приказ»;
- «Код работника»\* для сущности «Работник детского сада»;
- «Номер договора»\* для сущности «Договор»;
- Атрибуты «Дата», «Ребёнок» и «Группа» для сущности «Журнал»;
- «Код занятия»\* для «Дополнительного занятия»;
- «Код карты»\* для сущности «Медицинская карта».

Помеченные «звездочкой» первичные ключи являются суррогатными и нужны для придания первичному ключу нужных ему уникальности и компактности.

Для того чтобы создать единую информационную структуру, все таблицы базы данных необходимо объединить, создав для этого связи между их полями. Связанные таблицы дают возможность объединять все данные на основе совпадающих значений полей.

Связи устанавливаются между ключевыми атрибутами сущностей. В большинстве случаев с первичным ключом одной сущности, являющимся уникальным идентификатором каждой ее записи, связывается внешний ключ другой сущности.

Между сущностями могут быть установлены следующие типы связей:

Связь «один-к-одному» устанавливается в случаях, когда конкретная строка главной таблицы в любой момент времени связана только с одной строкой подчиненной таблицы. Эта связь самая простая, но встречается она нечасто. Таблицы, имеющие тип связи «один-к-одному», всегда можно скомпоновать в одну.

Связь «один-ко-многим» устанавливается в случаях, когда конкретная строка главной таблицы в любой момент времени связана с несколькими строками подчиненной таблицы. При этом любая строка подчиненной таблицы связана только с одной строкой главной таблицы.

Связь «многие-ко-многим» устанавливается в случаях, когда конкретная строка главной таблицы в любой момент времени связана с

несколькими строками подчиненной таблицы. Фактически такой тип связи – сложный и запутанный, и в базах данных его создавать не разрешается.

В данном случае, наша предметная область подразумевает следующие связи:

Связь «один-к-одному» между сущностями «Ребёнок» и «Медицинская карта» и между «Ребёнок» и «Договор». При поступлении ребенка в детский сад на него заключается договор и заводится мед. карта. На одного ребенка – одна мед. карта и один договор. Т.е. связь «один-к-одному».

Связь «один-ко-многим» между сущностями «Ребёнок» и «Приказ». В приказе может быть указан лишь один ребенок, но на одного ребенка может заводиться несколько приказов.

Связь «один-ко-многим» между сущностями «Группа» и «Приказ». В приказе устанавливается перевод в одну определенную группу, но таких приказов на перевод может быть много.

Один ребенок может посещать несколько дополнительных занятий, а на дополнительное занятие может ходить несколько детей. Это связь «многие-ко-многим».

Одно дополнительное занятие может вести один педагог (несколько во времени), один педагог не может вести несколько занятий. Это связь «один-ко-многим».

Сущности «Работник ДС» и «Группа» связаны связью «многие-ко-многим», т.к. в группе могут работать несколько работников, и этих работников могут перевести в другие группы (один работник работал в разных группах).

«Журнал» хранит информацию о каждом отдельно взятом дне. Между сущностями «Журнал» и «Группа» связь «один-ко-многим», т.к. журнал однозначно привязан к группе, но записей в журнале, относящихся к группе – много. То же самое со связью между «Журналом» и «Ребёнком». На одного ребёнка – множество записей в журнале, но конкретная запись связана лишь с одним ребенком. Связь «один-ко-многим».

Между «Журналом» и «Работником ДС» так же есть связь «один-многим». Каждую конкретную запись оставляет определенный работник, но он ответственный за более чем одну запись.

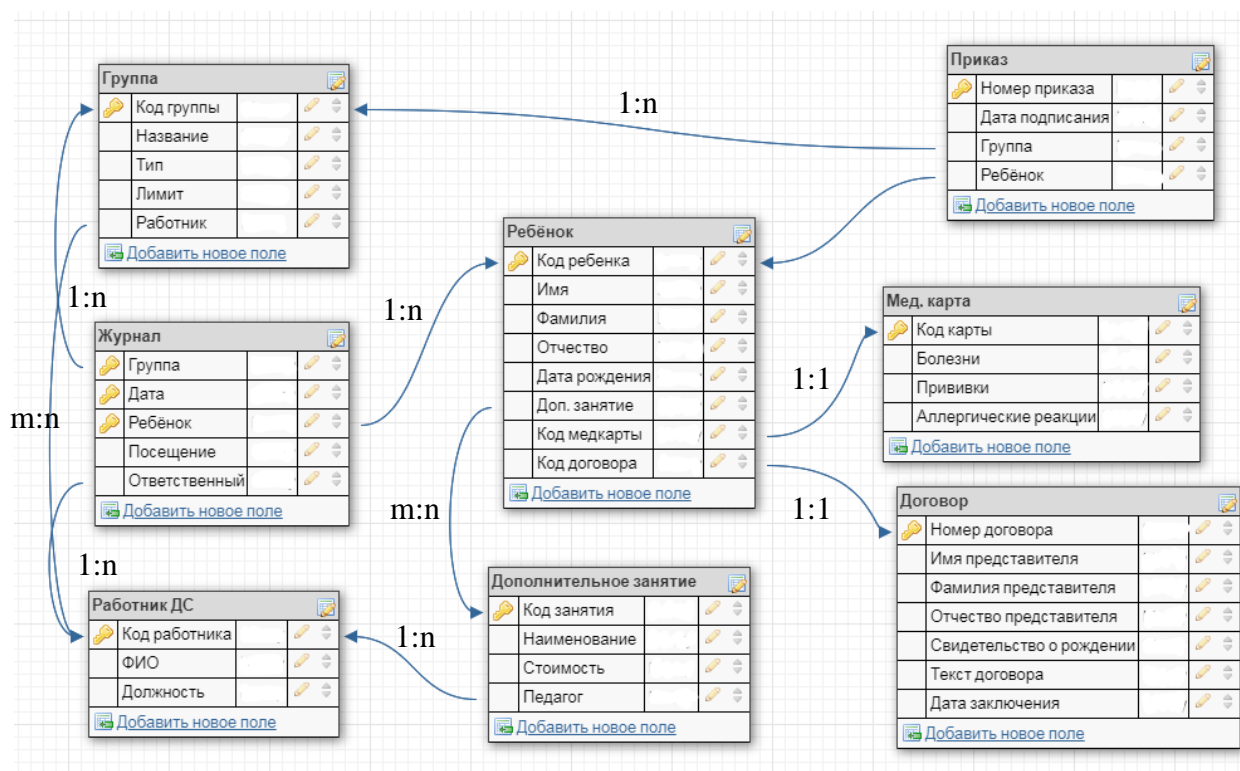


Рисунок 2.3 – Начальная логическая схема

## 2.3 Нормализация таблиц

Следующий шаг на пути проектирования структуры реляционной базы данных – нормализация таблиц. Нормализация — это процесс организации данных в базе данных, включающий создание таблиц и установление отношений между ними в соответствии с правилами, которые обеспечивают защиту данных и делают базу данных более гибкой, устраняя избыточность и несогласованные зависимости.

Нормализация таблиц представляет собой последовательное изменение структуры таблиц до тех пор, пока она не будет удовлетворять требованиям последней формы нормализации. Всего существует шесть форм нормализации: первая нормальная форма, вторая нормальная форма, третья

нормальная форма, нормальная форма Бойса-Кодда, четвертая нормальная форма и пятая нормальная форма или нормальная форма проекции-соединения. Однако концептуально важными являются только первые три нормальные формы, которые предложил Эдгар Кодд.

**Первая нормальная форма:** запрещает повторяющиеся столбцы (содержащие одинаковую по смыслу информацию); запрещает множественные столбцы (содержащие значения типа списка и т.п.); требует определить первичный ключ для таблицы, то есть тот столбец или комбинацию столбцов, которые однозначно определяют каждую строку, а также не допускает пустых значений в полях.

В таблице «Группа» есть повторяющееся атрибут «Тип группы» и «Лимит», при этом оба атрибута зависят друг от друга. Для нормализации этой таблицы необходимо разбить её на две таблицы: «Группа» и «Тип группы».

Таблица «Группа» хранит информацию о самой группе и имеет следующие атрибуты: «Код группы», «Название», «Код типа». «Код группы» является первичным ключом.

Таблица «Тип группы» хранит информацию о направленности группы и связанным с этим лимитом: «Код типа», «Наименование», «Лимит». Атрибут «Код типа» является первичным ключом и связывает данную таблицу с таблицей «Группа». Тип связи «один-ко-многим», т.к. групп одного и того же типа может быть много.

В таблице «Работник ДС» есть повторяющийся атрибут «Должность». Для нормализации разбиваем таблицу на 2 таблицы: «Работник ДС» и «Должность».

Таблица «Работник ДС» имеет следующий атрибуты: «Код работника», «ФИО», «Код должности». «Код работника» является первичным ключом.

Таблица «Должность» имеет следующие атрибуты: «Код должности», «Должность». Первичным атрибутом является «Код должности» и он



связывает таблицы «Работник ДС» и «Должность» связью «один-ко-многим», т.к. одна и та же должность может быть у нескольких работников.

Таблица «Мед. карта» состоит из атрибутов: «Код мед. карты», «Прививки», «Заболевания» и «Аллергические реакции». Данная таблица состоит из повторяющихся атрибутов, которые стоит вынести в отдельные таблицы. Таким образом таблица «Мед. карта» исчерпает себя и больше не понадобится.

Т.е. таблица «Сделанные прививки» будет иметь атрибуты: «Код прививки», «Название прививки», «Дата». Один ребенок может сделать несколько прививок, а одна по типу прививка может быть у нескольких детей. Это связь «многие-ко-многим», при этом в таблице остались повторяющиеся атрибуты. Для раскрытия связи «многие-ко-многим» создается вспомогательная таблица. Она будет носить название «Сделанная прививка», а предыдущая таблица изменит свое название на просто «Прививка».

Таблица «Прививка» будет состоять из атрибутов «Код прививки», «Наименование». «Код прививки» будет первичным ключом.

Вспомогательная таблица «Сделанная прививка» будет иметь атрибуты «Код прививки», «Код ребенка», «Дата». Все три атрибута будут являться первичным ключом.

Между таблицами «Прививка» и «Сделанная прививка» будет связь «один-ко-многим», как и между таблицами «Сделанная прививка» и «Ребенок».

«Заболевание» тоже стоит вынести отдельной таблицей. Т.к. сами заболевания тоже являются повторяющимся атрибутом (ребенок может болеть несколькими болезнями, так и одной болезнью может заболеть несколько детей), поэтому между таблицами «Ребёнок» и «Заболевание» наблюдается связь «многие-ко-многим». Требуется дополнительная таблица «Перенесенное заболевание».

Таблица «Заболевание» состоит из атрибутов «Код заболевания», «Наименование». «Код заболевания» — первичный ключ.

Таблица «Перенесенное заболевание» состоит из следующих атрибутов: «Код заболевания», «Код ребенка», «Начало болезни», «Окончание болезни», «Осложнения». Атрибуты «Код заболевания», «Код ребенка», «Начало болезни» являются первичным ключом.

Таблица «Аллергическая реакция» состоит атрибутов «Код аллергии», «Наименование». Здесь, как и в предыдущих рассмотренных таблицах связь «многие-ко-многим», а значит требуется дополнительная таблица «Аллергии ребенка».

Таблица «Аллергия ребенка» состоит из атрибутов «Код аллергии», «Код ребенка». Оба атрибута являются первичным ключом. Связь с таблицами «Аллергическая реакция», «Ребёнок» — «один-ко-многим».

Между таблицами «Ребёнок» и «Договор» существует связь «один-к-одному». Как уже говорилось ранее, в описании связи, данным таблицы можно скомпилировать в одну. Таким образом в таблицу «Ребёнок» добавятся следующие атрибуты: «Текст договора», «Дата заключения договора», «Свидетельство о Рождении».

Отдельно стоит вынести таблицу «Представитель». Её мы вынесли из убранной таблицы «Договор». Данная таблица будет содержать следующие атрибуты: «Код представителя», «Фамилия», «Имя», «Отчество», «Телефон», «Адрес». Первичным ключом является «Код представителя». В таблицу «Ребёнок» стоит добавить атрибут «Представитель», для того, чтобы связать таблицы «Ребёнок» и «Представитель» связью «один-ко-многим» (у нескольких детей может быть один представитель).

Между таблицами «Работник ДС» и «Группой» обнаруживается связь «многие-ко-многим». В группе работают несколько работников, и работники в разный момент времени могут работать в разных группах.

Для решения данной проблемы создаем новую вспомогательную таблицу «Работает на месте». Первичным ключом данной таблицы будет

совокупность атрибутов «Код работника» и «Время назначения». Атрибут «Код группы» указывает на конкретное место, в котором работает сотрудник. Этот атрибут связан с таблицами «Группа» соответственно. Атрибут «Дата снятия» хранит информацию о том, когда сотрудника сняли с места или перевели на новое.

Теперь подробнее рассмотрим получившуюся схему (рисунок 2.4) на соответствие следующим нормальным формам:

**Вторая нормальная форма.** Отношение находится во второй нормальной форме, если оно находится в первой нормальной форме и каждый неключевой атрибут зависит от всего первичного ключа (не зависит от части ключа). В случаях, когда таблица находится в первой нормальной форме и первичный ключ у нее состоит из одного столбца, то она автоматически находится во второй нормальной форме.

Проанализировав получившуюся схему можно сделать вывод, что таблицы базы данных «Учет детей в детском саду» находятся во второй нормальной форме.

**Третья нормальная форма.** Отношение находится в третьей нормальной форме, если оно находится во второй нормальной форме и если все неключевые атрибуты отношения взаимно независимы и полностью зависят от первичного ключа. При соблюдении данного условия, база данных соответствует требованиям третьей нормальной формы.

В ходе анализа, было выявлено, что вся база данных «Учет детей в детском саду» была приведена к третьей нормальной форме.



Рисунок 2.4 — Логическая схема после нормализации

## 2.4 Разработка физической схемы базы данных

Перед тем, как начинать разрабатывать физическую схему данных, стоит описать бизнес-правила.

Бизнес-правила представляют собой механизмы управления БД и предназначены для поддержания БД в целостном состоянии, а также для выполнения ряда других действий, например, накапливания статистики работы с БД. Многие бизнес-правила реализуются с помощью триггеров.

В данном случае нам потребуется реализовать следующие ограничения:

- детей в конкретной группе не должно быть больше, чем указано в её лимите;
- дата снятия с должности не должна быть меньше даты назначения;
- дата начала заболевания не должна быть больше даты выздоровления.

После построения логической и концептуальной моделей базы данных необходимо построить физическую модель базы данных с помощью модели данных конкретной СУБД. Разные СУБД поддерживают различные модели структуры баз данных. Существует немного подобных моделей БД, и наиболее популярной из них является реляционная.

Построение физической схемы базы данных является конечным этапом проектирования базы данных.

Прежде чем начать создавать физическую схему, необходимо определиться с выбором СУБД. Существует множество СУБД, таких как SQLite, PostgreSQL, MySQL и другие.

Для реализации данной базы данных была выбрана СУДБ MySQL, так как она имеет множество преимуществ, например:

- Простота в работе;
- Богатый функционал;
- Безопасность;

- Масштабируемость;
- Высокая скорость работы.

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.

SQL – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, управляемой соответствующей системой управления базами данных.

Для построения физической схемы базы данных создадим таблицы, которые отражают сущности:

- Таблицу Child\_card сущность Ребёнок;
- Таблицу Group сущность Группа;
- Таблицу Inoculation сущность Прививки;
- Таблицу Inoculation\_made сущность Сделанные прививки;
- Таблицу Delegate сущность Представитель;
- Таблицу Disease сущность Заболевания;
- Таблицу Child\_disease сущность Перенесенные заболевания;
- Таблицу Allergy сущность Аллергическая реакция;
- Таблицу Child\_Allergy сущность Аллергия ребёнка;
- Таблицу Order сущность Приказ;
- Таблицу Worker сущность Работник ДС;
- Таблицу Worker\_place сущность Работает на месте;
- Таблицу Role сущность Должность;
- Таблицу Add\_lesson сущность Дополнительное занятие;
- Таблицу Child\_lesson сущность Ребёнок на занятии;
- Таблицу Type сущность Тип группы;
- Таблицу Journal сущность Журнал.

Между таблицами построены связи, соответствующие описанным выше.



Данные связи были определены ещё при нормализации логической модели и с тех пор не поменялись, т.к. при нормализации мы уже избавились от связи «многие-ко-многим».

Для работы с СУБД так же стоит определить типы данных. Были выбраны следующие:

Int - длинное целое число. Используется для первичных ключей каждой таблицы. Так же этот тип данных используется внешних ключей. В нашей базе данных данным типом обладают все ID и связанные с ними внешние ключи, а так же цена Дополнительного занятия «Price».

Varchar(x) - символьные данные переменной длины, где x обозначает количество символов. Используется для хранения значений атрибутов, таких как «Name», «Surname», «Address», «Telephone» и им подобным.

Date используется для хранения даты в полном виде. В нашей база данных это следующие поля: «In\_date», «Date\_of\_birth», «ST\_date», «SP\_date» и другие.

Boolean используется в поле «Visit», для отметки о посещаемости.

Text используется для хранения текста, например «Contract\_Text».

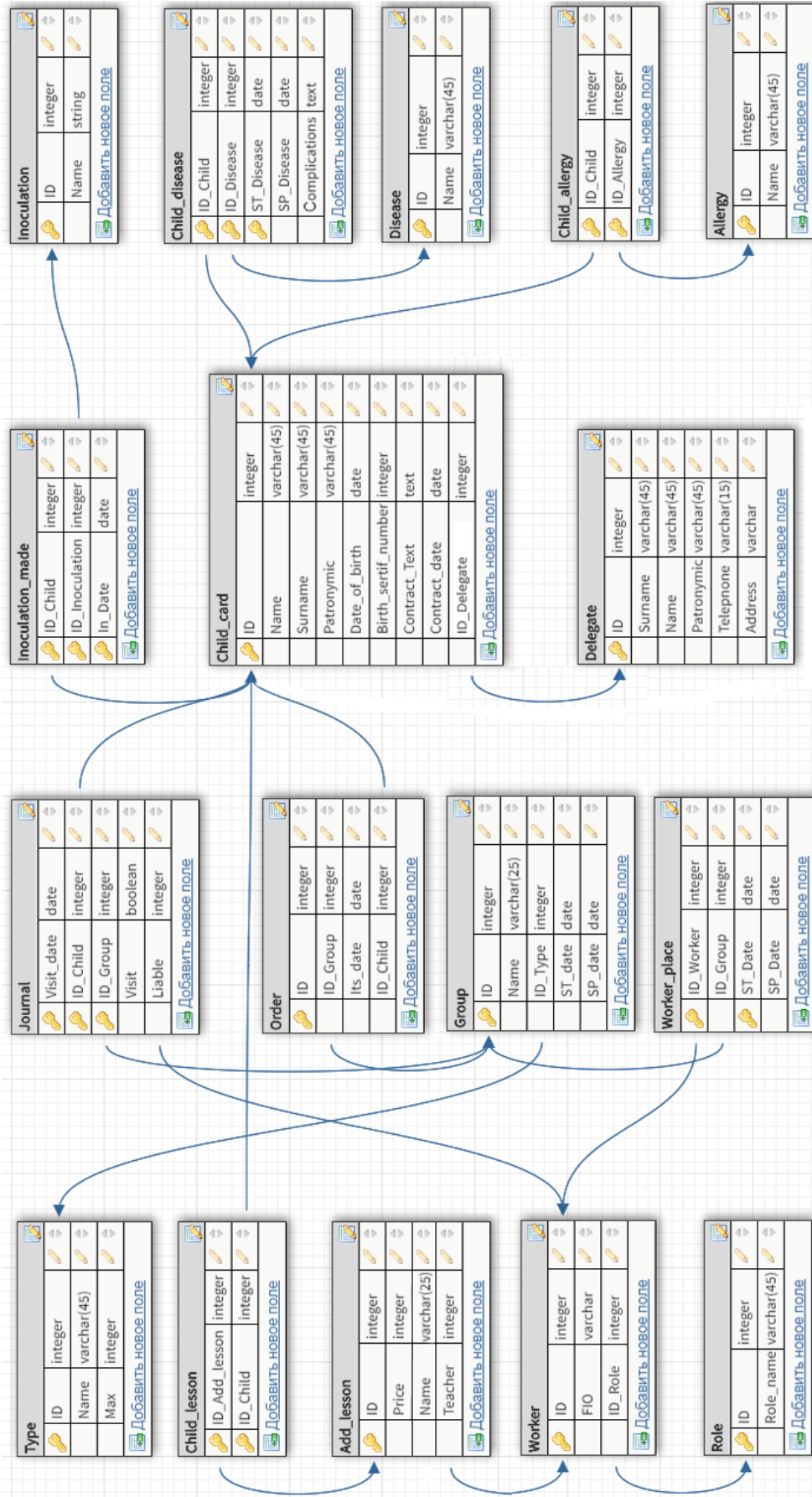


Рисунок 2.5 – Физическая схема БД

### 3 Реализация БД

Перед реализацией базы данных, стоит отметить, что не все имеющиеся название таблиц можно использовать в СУБД. Некоторые названия ( например «Group») зарезервированы самой СУБД. Поэтому их пришлось переименовать.

Скрипт создания таблицы «Child\_card» приведен ниже, остальные скрипты созданий таблиц приведены в приложении А.

```
CREATE TABLE `Child_card` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Name` varchar(45) NOT NULL,
  `Surname` varchar(45) NOT NULL,
  `Patronymic` varchar(45),
  `Date_of_birth` DATE NOT NULL,
  `Birth_sertif_number` INT NOT NULL UNIQUE,
  `Contract_Text` TEXT NOT NULL,
  `Contract_date` DATE NOT NULL,
  PRIMARY KEY (`ID`)
);
```

После создания таблиц стоит создать нужные нам связи. Для этого используется скрипт, приведенный ниже. Остальные скрипты создания внешних ключей приведены в Приложении А.

```
ALTER TABLE `Group` ADD CONSTRAINT `Group_fk0` FOREIGN
KEY (`ID_Type`) REFERENCES `Type`(`ID`);
```

Пример заполнения таблиц приведен на заполнении таблицы «Role» Скрипты заполнения остальных таблиц приведены в приложении А.

```
insert into Role values(default, 'Воспитатель'),
(default, 'Помощник воспитателя'),
(default, 'Педагог');
```

Заполненная таблица представлена на рисунке 3.1.

ID	Role_name
1	Воспитатель
2	Помошник воспитателя
3	Педагог
NULL	NULL

Рисунок 3.1 – Результат заполнения таблицы

### 3.1 Построение запросов к базе данных с помощью языка SQL

База данных необходима для обработки информации, одной из главных ее задач является обработка запросов и предоставление интересующей информации пользователю, избегая ненужные записи.

Для обращения к базе данных используются запросы, написанные на языке SQL. Запросом называется команда, которая передается серверу базы данных, и которая сообщает ему, что нужно вывести определенную информацию из таблиц в память. Эта информация обычно посылается непосредственно на экран компьютера или терминала или представить как вводную информацию для другой команды или процесса.

Например, нам нужно реализовать запрос на получении информации о работниках и группе, к которой они прикреплены. Т.к. в таблице «Worker\_place» информация прикреплена в виде внешних ключей, удобно будет совершить левое соединение таблиц «Worker\_place», «Worker», «Group», «Role». Информация будет отсортирована по Group.Name.

Скрипт запроса о предоставлении информации обо всех работниках групп представлен ниже. Скрипты остальных запросов представлены в приложении Б.

```
SELECT  DISTINCT   group.name,   worker.fio,   rolle.role_name,
worker_place.ST_Date
FROM worker_place
LEFT JOIN worker ON worker.id = worker_place.ID_worker
LEFT JOIN group ON group.id = worker_place.ID_group
LEFT JOIN rolle ON rolle.ID = worker.ID_role
ORDER BY group.name;
```

name	fio	role_name	ST_Date
Лютики	Иванова Мария Ивановна	Воспитатель	2010-09-01
Лютики	Петрова Ольга Ивановна	Воспитатель	2011-09-01
Лютики	Кравчева Мария Ивановна	Помошник воспитателя	2011-09-01
Ромашки	Семёнова Мария Сергеевна	Воспитатель	2011-09-01
Ромашки	Котова Дарья Ивановна	Воспитатель	2011-09-01
Ромашки	Иванова Галина Ивановна	Помошник воспитателя	2011-09-01

Рисунок 3.2 – Результат выполнения запроса

### 3.2 Разработка хранимых процедур и триггеров

Хранимые процедуры представляют собой набор команд SQL, которые могут компилироваться и храниться на сервере. Таким образом, вместо того, чтобы хранить часто используемый запрос, клиенты могут ссылаться на соответствующую хранимую процедуру. Это обеспечивает лучшую производительность, поскольку данный запрос должен анализироваться только однажды и уменьшается трафик между сервером и клиентом.

Триггер представляет собой хранимую процедуру, которая активизируется при наступлении определенного события.

Ниже представлена процедура, которая выводит список детей, у которых нет вводимой прививки:

```
CREATE PROCEDURE no_inoculat(IN Idi INT)
BEGIN
SELECT child_card.surname, child_card.name, child_card.patronymic
FROM child_card
WHERE child_card.ID not in (select inoculation_made.ID_child from
inoculation_made where inoculation_made.ID_inoculation = Idi);
```

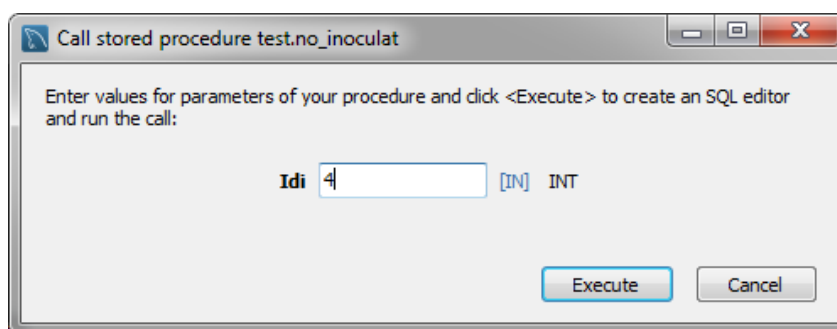


Рисунок 3.3 – Вызов процедуры

	surname	name	patronymic
	Иванков	Игорь	Иванович

Рисунок 3.4 – Результат выполнения процедуры

Список остальных хранимых процедур находится в приложении В.

Для ограничения целостности, а так же реализации некоторых бизнес-правил используют триггеры. Пример триггера, который проверяет дату снятия с должности на соответствие, представлен ниже. Описание всех триггеров находится в приложении В.

```
CREATE DEFINER=`root`@`localhost` TRIGGER
`worker_place_BEFORE_UPDATE` BEFORE UPDATE ON `worker_place`
FOR EACH ROW BEGIN
if (select worker_place.ST_date from worker_place
where ((worker_place.SP_date is null))>new.SP_date )
then SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Неправильная дата
снятия с должности';
END if;
END
```



## **Заключение**

В результате выполнения курсовой работы была достигнута поставленная цель, выполнены поставленные задачи: изучена работа детского сада, построены концептуальная, логическая и физическая схемы базы данных, так же построена база данных в СУБД(MySQL), разработаны запросы, триггеры и процедуры для базы данных.

## Список использованной литературы

1. Введение в триггеры MySql - RUSELLER.COM [Электронный ресурс]. – Режим доступа: <https://ruseller.com/lessons.php/>\_, свободный. – Загл. с экрана
2. Нормальная форма - Википедия [Электронный ресурс]. – Режим доступа: <https://ru.wikipedia.org/wiki/>\_, свободный. – Загл. с экрана.
3. Билл Карвин. Программирование баз данных SQL. Типичные ошибки и их устранение / Карвин Билл. - : Рид Групп, 2011. – 336 с.
4. Томас Коннолли. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Коннолли Томас, Бегг Каролин. - : Вильямс, 2017. – 1440 с.
5. Хомоненко А.Д., Цыганков В.М., Мальцев М.Г. - Базы данных. Учебник для высших учебных заведений (6-е изд.) / А.Д. Хомоненко , В.М. Цыганков , М.Г. Мальцев. - М.: КОРОНА-Век, 2009. - 322 с.

## Приложения А

```
CREATE TABLE `Child_card` (  
    `ID` INT NOT NULL AUTO_INCREMENT,  
    `Name` varchar(45) NOT NULL,  
    `Surname` varchar(45) NOT NULL,  
    `Patronymic` varchar(45),  
    `Date_of_birth` DATE NOT NULL,  
    `Birth_sertif_number` INT NOT NULL UNIQUE,  
    `Contract_Text` TEXT NOT NULL,  
    `Contract_date` DATE NOT NULL,  
    PRIMARY KEY (`ID`)  
);  
  
CREATE TABLE `Group` (  
    `ID` INT NOT NULL AUTO_INCREMENT,  
    `Name` varchar(25) NOT NULL,  
    `ID_Type` INT NOT NULL,  
    `ST_date` DATE NOT NULL,  
    `SP_date` DATE,  
    PRIMARY KEY (`ID`)  
);  
  
CREATE TABLE `Journal` (  
    `Visit_date` DATE NOT NULL,  
    `ID_Child` INT NOT NULL,  
    `ID_Group` INT NOT NULL,  
    `Visit` BOOLEAN NOT NULL,  
    `Liable` INT NOT NULL,  
    PRIMARY KEY (`Visit_date`, `ID_Child`, `ID_Group`)
```

);

```
CREATE TABLE `Worker` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `FIO` varchar(255) NOT NULL,
    `ID_Role` INT NOT NULL,
    PRIMARY KEY (`ID`)
);
```

```
CREATE TABLE `Add_lesson` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `Name` varchar(45) NOT NULL,
    `Price` INT NOT NULL,
    `Teacher` INT NOT NULL,
    PRIMARY KEY (`ID`)
);
```

```
CREATE TABLE `Child_lesson` (
    `ID_Add_lesson` INT NOT NULL,
    `ID_Child` INT NOT NULL,
    PRIMARY KEY (`ID_Add_lesson`,`ID_Child`)
);
```

```
CREATE TABLE `Type` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `Name` varchar(45) NOT NULL,
    `Max` INT NOT NULL,
    PRIMARY KEY (`ID`)
);
```

```
CREATE TABLE `Rolle` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Role_name` varchar(45) NOT NULL,
  PRIMARY KEY (`ID`)
);
```

```
CREATE TABLE `Inoculation` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Name` VARCHAR(255) NOT NULL,
  PRIMARY KEY (`ID`)
);
```

```
CREATE TABLE `Inoculation_made` (
  `ID_Child` INT NOT NULL,
  `ID_Inoculation` INT NOT NULL,
  `In_Date` DATE NOT NULL,
  PRIMARY KEY (`ID_Child`,`ID_Inoculation`,`In_Date`)
);
```

```
CREATE TABLE `Disease` (
  `ID` INT NOT NULL AUTO_INCREMENT,
  `Name` varchar(45) NOT NULL,
  PRIMARY KEY (`ID`)
);
```

```
CREATE TABLE `Child_disease` (
  `ID_Child` INT NOT NULL,
  `ID_Disease` INT NOT NULL,
  `ST_Disease` DATE NOT NULL,
  `SP_Disease` DATE,
```

```

        `Complications` TEXT,
        PRIMARY KEY (`ID_Child`, `ID_Disease`, `ST_Disease`)
    );

```

```

CREATE TABLE `Allergy` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `Name` varchar(45) NOT NULL,
    PRIMARY KEY (`ID`)
);

```

```

CREATE TABLE `Child_allergy` (
    `ID_Child` INT NOT NULL,
    `ID_Allergy` INT NOT NULL,
    PRIMARY KEY (`ID_Child`, `ID_Allergy`)
);

```

```

CREATE TABLE `Delegatte` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `Surname` varchar(45) NOT NULL,
    `Name` varchar(45) NOT NULL,
    `Patronymic` varchar(45) NOT NULL,
    `Telephone` varchar(15) NOT NULL,
    `Address` varchar(255) NOT NULL,
    `ID_Child` INT NOT NULL,
    PRIMARY KEY (`ID`)
);

```

```

CREATE TABLE `Orderr` (
    `ID` INT NOT NULL AUTO_INCREMENT,
    `ID_Groupp` INT NOT NULL,

```



```

        `Its_date` DATE NOT NULL,
        `ID_Child` INT NOT NULL,
        PRIMARY KEY (`ID`)
    );

```

```

CREATE TABLE `Worker_place` (
    `ID_Worker` INT NOT NULL,
    `ID_Groupp` INT NOT NULL,
    `ST_Date` DATE NOT NULL,
    `SP_Date` DATE,
    PRIMARY KEY (`ID_Worker`, `ST_Date`)
);

```

```

ALTER TABLE `Groupp` ADD CONSTRAINT `Groupp_fk0` FOREIGN KEY
(`ID_Type`) REFERENCES `Type`(`ID`);

```

```

ALTER TABLE `Journal` ADD CONSTRAINT `Journal_fk0` FOREIGN KEY
(`ID_Child`) REFERENCES `Child_card`(`ID`);

```

```

ALTER TABLE `Journal` ADD CONSTRAINT `Journal_fk1` FOREIGN KEY
(`ID_Groupp`) REFERENCES `Groupp`(`ID`);

```

```

ALTER TABLE `Journal` ADD CONSTRAINT `Journal_fk2` FOREIGN KEY
(`Liable`) REFERENCES `Worker`(`ID`);

```

```

ALTER TABLE `Worker` ADD CONSTRAINT `Worker_fk0` FOREIGN KEY
(`ID_Role`) REFERENCES `Rolle`(`ID`);

```

```

ALTER TABLE `Add_lesson` ADD CONSTRAINT `Add_lesson_fk0` FOREIGN
KEY (`Teacher`) REFERENCES `Worker`(`ID`);

```

```
ALTER TABLE `Child_lesson` ADD CONSTRAINT `Child_lesson_fk0`  
FOREIGN KEY (`ID_Add_lesson`) REFERENCES `Add_lesson`(`ID`);
```

```
ALTER TABLE `Child_lesson` ADD CONSTRAINT `Child_lesson_fk1`  
FOREIGN KEY (`ID_Child`) REFERENCES `Child_card`(`ID`);
```

```
ALTER TABLE `Inoculation_made` ADD CONSTRAINT  
`Inoculation_made_fk0` FOREIGN KEY (`ID_Child`) REFERENCES  
`Child_card`(`ID`);
```

```
ALTER TABLE `Inoculation_made` ADD CONSTRAINT  
`Inoculation_made_fk1` FOREIGN KEY (`ID_Inoculation`) REFERENCES  
`Inoculation`(`ID`);
```

```
ALTER TABLE `Child_disease` ADD CONSTRAINT `Child_disease_fk0`  
FOREIGN KEY (`ID_Child`) REFERENCES `Child_card`(`ID`);
```

```
ALTER TABLE `Child_disease` ADD CONSTRAINT `Child_disease_fk1`  
FOREIGN KEY (`ID_Disease`) REFERENCES `Disease`(`ID`);
```

```
ALTER TABLE `Child_allergy` ADD CONSTRAINT `Child_allergy_fk0`  
FOREIGN KEY (`ID_Child`) REFERENCES `Child_card`(`ID`);
```

```
ALTER TABLE `Child_allergy` ADD CONSTRAINT `Child_allergy_fk1`  
FOREIGN KEY (`ID_Allergy`) REFERENCES `Allergy`(`ID`);
```

```
ALTER TABLE `Delegatte` ADD CONSTRAINT `Delegate_fk0` FOREIGN  
KEY (`ID_Child`) REFERENCES `Child_card`(`ID`);
```

```
ALTER TABLE `Orderr` ADD CONSTRAINT `Order_fk0` FOREIGN KEY
(`ID_Groupp`) REFERENCES `Groupp`(`ID`);
```

```
ALTER TABLE `Orderr` ADD CONSTRAINT `Order_fk1` FOREIGN KEY
(`ID_Child`) REFERENCES `Child_card`(`ID`);
```

```
ALTER TABLE `Worker_place` ADD CONSTRAINT `Worker_place_fk0`
FOREIGN KEY (`ID_Worker`) REFERENCES `Worker`(`ID`);
```

```
ALTER TABLE `Worker_place` ADD CONSTRAINT `Worker_place_fk1`
FOREIGN KEY (`ID_Groupp`) REFERENCES `Groupp`(`ID`);
```

### **Скрипты заполнения базы данных:**

```
insert into Rolle values(default, 'Воспитатель'),
    (default, 'Помощник воспитателя'),
    (default, 'Педагог');
```

```
insert into Worker values(default, 'Иванова Марья Ивановна', 1),
    (default, 'Петрова Ольга Ивановна', 1),
    (default, 'Семёнова Марья Сергеевна', 1),
    (default, 'Котова Дарья Ивановна', 1),
    (default, 'Каучук Степан Михайлович', 3),
    (default, 'Кравчева Марья Ивановна', 2),
    (default, 'Иванова Галина Ивановна', 2);
```

```
insert into Add_lesson values(default, 'Лепка', 500, 5);
```

```
insert into Type values(default, 'Общеразвивающая', 35),
    (default, 'Офтольмологическая', 20),
    (default, 'Логопедическая', 20);
```

```
insert into Groupp values(default, 'Лютики', 1, '2010-09-01', '2015-06-01'),
    (default, 'Ромашки', 2, '2010-09-01', '2015-06-01');
```

```
insert into worker_place values(1,1, '2010-09-01', null),
    (2,1, '2010-09-01', null),
    (6,1, '2010-09-01', null),
    (3,2, '2010-09-01', null),
    (4,2, '2010-09-01', null),
    (7,2, '2010-09-01', null);
```

```
insert into delegatte
values(default,'Иванова','Ольга','Викторовна','89912391222','где-то'),
    (default,'Катькин','Олег','Викторович','89912391222','где-то'),
    (default,'Петрова','Ольга','Викторовна','89912391222','где-то'),
    (default,'Семенов','Игорь','Викторович','89912391222','где-то'),
    (default,'Иванкова','Ольга','Викторовна','89912391222','где-то');
```

```
insert into child_card values(default,'Иван','Иванов','Иванович','2009-03-
04',54123,',бла-бла-бла','2010-08-12',1),
    (default,'Петр','Иванов','Иванович','2009-03-04',54122,',бла-бла-
бла','2010-08-12',1),
    (default,'Екатерина','Катькина','Ивановна','2008-12-04',54124,',бла-бла-
бла','2010-08-10',2),
    (default,'Николай','Петров','Иванович','2009-05-04',54125,',бла-бла-
бла','2010-07-26',3),
    (default,'Иван','Семенов','Иванович','2009-03-04',54127,',бла-бла-бла','2010-
08-12',4),
    (default,'Игорь','Иванков','Иванович','2009-03-04',54121,',бла-бла-бла','2010-
08-12',5);
```

```
insert into Orderr values(default, 1, '2010-08-12',1),
    (default, 1, '2010-08-12',2),
    (default, 1, '2010-08-10',3),
    (default, 2, '2010-08-12',4),
    (default, 2, '2010-08-12',5),
    (default, 2, '2010-08-12',6);
```

```
insert into Journal values('2010-09-01',1,1, true,1),
    ('2010-09-02',1,1, true, 2),
    ('2010-09-03',1,1, false, 1),
    ('2010-09-01',2,1, true,1),
    ('2010-09-02',2,1, true, 2),
    ('2010-09-03',2,1, true, 1),
    ('2010-09-01',3,1, false,1),
    ('2010-09-02',3,1, true, 2),
    ('2010-09-03',3,1, true, 1),
    ('2010-09-01',4,2, true, 3),
    ('2010-09-02',4,2, true, 4),
    ('2010-09-03',4,2, false, 3),
    ('2010-09-01',5,2, true, 3),
    ('2010-09-02',5,2, true, 4),
    ('2010-09-03',5,2, false, 3),
    ('2010-09-01',6,2, true, 3),
    ('2010-09-02',6,2, true, 4),
    ('2010-09-03',6,2, false, 3);
```

```
insert into child_lesson values(1,3);
```

```
insert into inoculation values(default,'От гриппа'),
    (default,'От столбняка'),
```

```
(default,'От дифтерии'),  
(default,'От туберкулеза');
```

```
insert into inoculation_made values(1,4,'2010-03-04'),  
  (2,4,'2010-03-08'),  
  (3,4,'2009-12-05'),  
  (4,4,'2010-05-04'),  
  (5,4,'2010-03-02'),  
  (1,4,'2010-03-01');
```

```
insert into disease values(default, 'ангина');
```

```
insert into child_disease values(4,1,'2010-09-03',null,null);
```

```
insert into allergy values(default,'На витамин С'),  
  (default, 'На пыль');
```

```
insert into child_allergy values(1,1),  
  (2,1),  
  (6,2);
```

## Приложение Б

### Скрипты запросов:

```
SELECT      DISTINCT      group.name,      worker.fio,      rolle.role_name,
worker_place.ST_Date
FROM worker_place
LEFT JOIN worker ON worker.id = worker_place.ID_worker
LEFT JOIN group ON group.id = worker_place.ID_group
LEFT JOIN rolle ON rolle.ID = worker.ID_role;
```

```
SELECT      DISTINCT      child_card.surname,      child_card.name,
child_card.Patronymic, add_lesson.name, worker.FIO, add_lesson.price
FROM child_lesson
LEFT JOIN child_card ON child_card.ID = child_lesson.ID_Child
LEFT JOIN add_lesson ON add_lesson.ID = child_lesson.ID_Add_lesson
LEFT JOIN worker ON worker.ID = add_lesson.teacher
ORDER BY child_card.surname;
```

```
SELECT DISTINCT type.name, group.name, worker.fio, rolle.role_name
FROM type
LEFT JOIN group ON type.ID = group.ID_Type
LEFT JOIN worker_place ON group.ID = worker_place.ID_Group
LEFT JOIN worker ON worker.ID = worker_place.ID_worker
LEFT JOIN rolle ON rolle.ID = worker.ID_Role
ORDER BY type.name;
```

```
SELECT Count(*) As child_count
From Orderr
Where Orderr.ID_Group=1;
```

## Приложение В

### Скрипты процедур:

1) Процедура, которая выводит количество детей в группе

```
CREATE PROCEDURE child_count(IN Idi INT)
```

```
BEGIN
```

```
    SELECT Count(*) As child_count
```

```
    From Orderr
```

```
    where Orderr.ID_Groupp=Idi;
```

```
END
```

2) Процедура, которая выводит детей, которые не сделали определенную прививку (по названию)

```
CREATE PROCEDURE no_inoculat(IN Idi varchar(255))
```

```
BEGIN
```

```
SELECT child_card.surname, child_card.name, child_card.patronymic
```

```
FROM child_card
```

```
WHERE child_card.ID not in (select inoculation_made.ID_child from
inoculation_made join inoculation on inoculation_made.ID_inoculation =
inoculation.ID where inoculation.Name = Idi);
```

3) Процедура, которая ищет детей по фамилии

```
CREATE PROCEDURE Find_Child(IN Surname varchar(45))
```

```
BEGIN
```

```
SELECT          DISTINCT          child_card.surname,          child_card.name,
child_card.Patronymic, groupp.name
```

```
FROM orderr
```

```
LEFT JOIN child_card ON child_card.ID = orderr.ID_Child
```

```
LEFT JOIN groupp ON groupp.ID = orderr.ID_Groupp
```



```
WHERE child_card.surname = surname;
END
```

4) Процедура, которая обновляет цену на дополнительные занятия

```
CREATE PROCEDURE New_count(IN ID INT, IN news INT)
BEGIN
    UPDATE Add_lesson SET Add_lesson.Price = news
    WHERE Add_lesson.ID = ID;
END
```

### **Скрипты триггеров:**

```
CREATE          DEFINER=`root`@`localhost`          TRIGGER
`worker_place_BEFORE_UPDATE` BEFORE UPDATE ON `worker_place`
FOR EACH ROW BEGIN
if (select worker_place.ST_date from worker_place
where ((worker_place.SP_date is null))>new.SP_date )
then SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Неправильная дата
снятия с должности';
END if;
END
```

```
CREATE          DEFINER=`root`@`localhost`          TRIGGER
`child_disease_BEFORE_UPDATE` BEFORE UPDATE ON `child_disease` FOR
EACH ROW BEGIN
if (select child_disease.ST_Disease from child_disease
where ((child_disease.SP_Disease is null))>new.SP_Disease)
then SIGNAL SQLSTATE '02000' SET MESSAGE_TEXT = 'Неправильная дата
окончания заболевания';
END if;
```

END

```
CREATE DEFINER=`root`@`localhost` TRIGGER `orderr_BEFORE_INSERT`  
BEFORE INSERT ON `orderr` FOR EACH ROW BEGIN  
    if ((select count(*) as limitt from orderr where orderr.ID_groupp =  
new.ID_groupp)+1)>(select Typepe.Max from Typepe join groupp  
        On groupp.ID_Typepe = Typepe.ID where new.ID_groupp =  
groupp.ID)  
        then SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Вы  
пытаетесь превысить лимит группы';  
    END IF;  
END
```