

Министерство науки и высшего образования РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
“Забайкальский государственный университет”
Энергетический факультет
Кафедра информатики и вычислительной техники

Курсовой проект

по предмету:

Базы данных

На тему:

**«Разработка реляционной базы данных по
выбранной предметной области (Сдача в аренду
торговых площадей)»**

Выполнил:

Студент(ФИО): Степанов Валерий Евгеньевич

Группа: ИВТ-19

Курс: Второй

Форма обучения: Очная

Руководитель:-

Должность: Доцент кафедры ИВТ и ПМ

ФИО: Гончаров Денис Сергеевич

г. Чита – 2021

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Факультет: Энергетический факультет
Кафедра: Информатики и вычислительной техники

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовой работе

По базам данных

На тему: создание базы данных для фирмы, занимающейся сдачей в аренду торговых площадей

Цель курсовой работы: Освоение принципов и методов создания, проектирования и обработки баз данных.

Выполнил студент группы ИВТ-19 Степанов Валерий Евгеньевич

Руководитель работы: Доцент кафедры ИВТ и ПМ Гончаров Денис Сергеевич

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Забайкальский государственный университет»
(ФГБОУ ВО «ЗабГУ»)
Факультет: Энергетический факультет Кафедра:
Информатики и вычислительной техники

ЗАДАНИЕ
на курсовую работу

По дисциплине базы данных

Студенту Степанову Валерию Евгеньевичу

специальности Информатика и Вычислительная техника

- 1) Тема курсовой работы: создание базы данных для фирмы, занимающейся сдачей в аренду торговых площадей.
- 2) Срок подачи студентом законченной работы: 09.06.2021 .
- 3) Исходные данные к работе: Рабочая сфера – фирма по продаже запчастей для автомобилей. Создать базу данных для учета работы с поставщиками.
- 4) Перечень подлежащих разработке в курсовой работе вопросов:

Схема БД, Нормализация БД, Ограничения целостности БД, Запросы к БД, Триггеры.

Дата выдачи задания: 25.02.2021

Руководитель курсовой работы _____ Задание
принял к исполнению

«1» мая 2021г.

Подпись студента _____ / _____ /

Календарный план
Выполнения курсового проекта

УТВЕРЖДАЮ

Зав. Кафедрой _____

«__» _____ 2021г.

Этапы выполнения курсовой работы	Месяцы и недели															
	Февраль				Март				Апрель				Май			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1.Получения задания на курсовую работу.		+	+													
2.Работа с источниками информации и подбор материала.				+	+											
3.Написание первого раздела.						+	+									
4.Написание второго раздела.								+	+							
5.Оформление работы в соответствии с методическими указаниями.										+	+					
6.Предоставление руководителю чернового варианта работы.												+				
7.Корректировка работы в соответствии с замечаниями.													+	+		
8.Защита работы															+	+

План выполнен: Руководитель _____

(подпись, расшифровка подписи)

«__» _____ 2021г.

Реферат

Курсовая работа по предмету базы данных, на тему: «Разработка реляционной базы данных по выбранной предметной области (фирма, занимающаяся продажей запасных частей для автомобилей)», студента 2-го курса Степанова В.Е.

Курсовая работа изложена на X страницах машинописного текста. Состоит из: Введения, 2-х глав, Заключения и списка литературы. Содержит вставки кода на языке запросов SQL, скриншоты результатов работы, таблиц для анализа и диаграммы “Сущность-Связь”

Цель работы: Приобретение навыков моделирования, разработки и работы с базой данных.

Ключевые слова: База данных, SQL, запрос, триггер, таблица, данные, анализ.

Содержание.

Введение.....	7
Глава 1. Теоретические сведения.....	8
Глава 2. Практическая часть.....	19
2.1 Описание предметной области.....	19
2.2 Диаграмма «Сущность-связь».....	20
2.3 Нормализация реляционной модели. Нормальные формы.....	21
2.4 Описание декларативных ограничений целостности.....	24
2.5 Реализация запросов.....	25
2.6 Реализация процедурных ограничений целостности.....	28
Заключение.....	31
Список используемых источников.....	32

Введение.

Базы данных — это совокупность структур, предназначенных для хранения больших объемов информации и программных модулей, осуществляющих управление данными, их выборку, сортировку и другие подобные действия.

Информация базы данных хранится в одной или нескольких таблицах. Любая таблица с данными состоит из набора однотипных записей, расположенных друг за другом. Они представляют собой строки таблицы, которые можно добавлять, удалять или изменять.

Каждая запись является набором именованных полей, или ячеек, которые могут хранить самую разнообразную информацию, начиная от даты рождения и заканчивая подробным описанием кулинарного рецепта. Однотипные поля разных записей образуют столбец таблицы.

Создав одну таблицу, вы уже получаете полноценную базу данных. Однако в реальной жизни структуры баз данных, а соответственно и способы их создания, намного сложнее.

В последние годы на первый план выдвигается новая отрасль - информационная индустрия, связанная с производством технических средств, методов, технологий для производства новых знаний. Эта индустрия тесно связана с развитием компьютерных технологий.

В информационном обществе доминирует производство информационного продукта, а материальный продукт становится более информационно емким. Изменяется весь уклад жизни, система ценностей: возрастает значимость культурного досуга, возрастает спрос на знания, от человека требуется способность к интеллектуальному труду и творчеству. В результате появились противоречия между ограниченными возможностями человека по восприятию и переработке информации и существующими массивами хранящейся и передаваемой информации.

Возникло большое число избыточной информации, в которой иногда трудно сориентироваться и выбрать нужные сведения.

Для решения подобных проблем применяются автоматизированные базы данных. Они стали неотъемлемой частью практически всех компьютерных систем - от отрасли до отдельного предприятия. За последние несколько лет вырос уровень потребительских качеств систем управления базами данных (СУБД): разнообразие поддерживаемых функций, удобный для пользователя интерфейс, сопряжение с программными продуктами, в частности с другими СУБД, возможности для работы в сети и т.д. СУБД позволяет сводить воедино информацию из самых разных источников (электронные таблицы, другие базы данных) и помогает быстро найти необходимую информацию, донести ее до окружающих с помощью отчетов, графиков или таблиц.

Базы данных являются неотъемлемой частью любого крупного проекта, так как все они нуждаются в хранении, обработке и изменении данных. Умение работать с базами данных сейчас необходимо, наверное, каждому человеку, и уж тем более программисту. В данной работе мы рассмотрим самые необходимые навыки для создания и работы с базой данных на примере базы данных для учета стоимости прошедшей в эфире рекламы.

Объект работы: Базы данных.

Предмет работы: База данных для учета стоимости прошедшей в эфире рекламы.

Целью данной курсовой работы является Приобретение навыков моделирования, разработки и работы с базой данных.

Глава 1. Теоретические сведения.

Данные — это представление фактов и идей в формализованном виде, пригодном для передачи и обработки в некотором информационном

процессе. Данные — это выделенная (из системы, благодаря обособленности существования носителя) информация.

База данных (по Дж. Мартину) — совокупность взаимосвязанных данных, совместно используемых несколькими приложениями и хранящимися с (минимальной) регулируемой избыточностью. Данные запоминаются таким образом, чтобы они, по мере возможности, не зависели от программ. Для обработки данных применяется общий управляющий метод доступа. Если базы данных не пересекаются по структуре, то говорят о системе баз данных.

Базы данных состоят из всех экземпляров записей, экземпляров наборов записей и областей, которые контролируются конкретной схемой. (Под схемой можно понимать карту всей логической структуры базы данных)

Свойства БД:

- структурная целостность;
- непротиворечивость (семантическая целостность);
- отсутствие избыточности.

Системой управления базами данных (**СУБД**) называется совокупность программных средств, необходимых для использования базы данных и предоставляющих разработчикам и пользователям множество различных представлений данных.

Модель данных — это инструмент представления концептуальной модели предметной области и динамики ее изменения в виде базы данных.

Первое, что мы создаем при разработке базы данных — это ее модель. Для этой работы нам понадобится специальное приложение. Одно из таких — **pgModeler**.

pgModeler — это свободный и открытый, кроссплатформенный инструмент визуального проектирования баз данных, объединяющий собой классические диаграммы сущность-связь с особенностями PostgreSQL.

Процесс проектирования БД представляет собой последовательность переходов от неформального словесного описания информационной структуры предметной области к формализованному описанию объектов предметной области в терминах некоторой модели. Выделяют следующие этапы проектирования:

1. Системный анализ и словесное описание информационных объектов предметной области.
2. Проектирование инфологической модели предметной области — частично формализованное описание объектов предметной области в терминах некоторой семантической модели, например, в терминах ER-модели.
3. Даталогическое или логическое проектирование БД, то есть описание БД в терминах принятой даталогической модели данных.
4. Физическое проектирование БД, то есть выбор эффективного размещения БД на внешних носителях для обеспечения наиболее эффективной работы приложения.

Создав модель нам нужно провести ее анализ. Для этого нам нужно провести нормализацию базы данных. В идеале результат будет ровно такой же, как схема созданная ранее.

Инфологическая модель должна включать такое формализованное описание предметной области, которое легко будет "читаться" не только специалистами по базам данных. И это описание должно быть настолько емким, чтобы можно было оценить глубину и корректность проработки проекта БД, и конечно, как говорилось раньше, оно не должно быть привязано к конкретной СУБД. Выбор СУБД — это отдельная задача, для корректного ее решения необходимо иметь проект, который не привязан ни к какой конкретной СУБД.

Модель "сущность — связь"

В основе ER-модели лежат следующие базовые понятия:

Сущность с помощью которой моделируется класс однотипных объектов. Сущность имеет имя, уникальное в пределах моделируемой системы. Предполагается, что в системе существует множество экземпляров данной сущности. Объект, которому соответствует понятие сущности, имеет свой набор атрибутов — характеристик, определяющих свойства данного представителя класса. Набор атрибутов, однозначно идентифицирующий конкретный экземпляр сущности, называют ключевым.

Между сущностями могут быть установлены **связи** — бинарные ассоциации, показывающие, каким образом сущности соотносятся или взаимодействуют между собой.

Связи делятся на три типа по множественности:

- один-к-одному (1:1),;
- один-ко-многим (1:M);
- многие-ко-многим (M:M).

Связь один-к-одному означает, что экземпляр одной сущности связан только с одним экземпляром другой сущности. Связь 1: M означает, что один экземпляр сущности, расположенный слева по связи, может быть связан с несколькими экземплярами сущности, расположенными справа по связи. Связь "один-к-одному" (1:1) означает, что один экземпляр одной сущности связан только с одним экземпляром другой сущности, а связь "многие-ко-многим" (M:M) означает, что один экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и наоборот, один экземпляр второй сущности может быть связан с несколькими экземплярами первой сущности.

Между двумя сущностями может быть задано сколько угодно связей с разными смысловыми нагрузками.

В ER-модели допускается принцип категоризации сущностей: вводится понятие подтипа сущности, то есть сущность может быть представлена в виде двух или более своих подтипов — сущностей, каждая из которых может иметь общие атрибуты и отношения и/или атрибуты и отношения, которые определяются однажды на верхнем уровне и наследуются на нижнем уровне. Все подтипы одной сущности рассматриваются как взаимоисключающие, и при разделении сущности на подтипы она должна быть представлена в виде полного набора взаимоисключающих подтипов.

Нормальные формы — это рекомендации по проектированию баз данных. Вы не обязаны придерживаться всех пяти нормальных форм при проектировании баз данных. Тем не менее, рекомендуется нормализовать базу данных в некоторой степени потому, что этот процесс имеет ряд существенных преимуществ с точки зрения эффективности и удобства обращения с вашей базой данных.

- 1) В нормализованной структуре базы данных вы можете производить сложные выборки данных относительно простыми SQL-запросами.
- 2) Целостность данных. Нормализованная база данных позволяет надежно хранить данные.
- 3) Нормализация предотвращает появление избыточности хранимых данных. Данные всегда хранятся только в одном месте, что делает легким процесс вставки, обновления и удаления данных. Есть исключение из этого правила. Ключи, сами по себе, хранятся в нескольких местах потому, что они копируются как внешние ключи в другие таблицы.

Масштабируемость — это возможность системы справляться с будущим ростом. Для базы данных это значит, что она должна быть способна работать быстро, когда число пользователей и объемы данных возрастают.

В теории реляционных баз данных обычно выделяется следующая последовательность нормальных форм:

- первая нормальная форма (1NF);
- вторая нормальная форма (2NF)(Переменная отношения находится во второй нормальной форме (2NF) тогда и только тогда, когда она находится в первой нормальной форме, и каждый неключевой атрибут минимально функционально зависит от первичного ключа);
- третья нормальная форма (3NF)(Переменная отношения находится в третьей нормальной форме (3NF) в том и только в том случае, когда она находится во второй нормальной форме, и каждый неключевой атрибут нетранзитивно⁴ функционально зависит от первичного ключа);
- нормальная форма Бойса-Кодда (BCNF)(Переменная отношения находится в нормальной форме Бойса-Кодда (BCNF) в том и только в том случае, когда любая выполняемая для этой переменной отношения нетривиальная и минимальная FD имеет в качестве детерминанта некоторый возможный ключ данного отношения);
- четвертая нормальная форма (4NF);
- пятая нормальная форма, или нормальная форма проекции-соединения (5NF или PJ/NF).

Основные свойства нормальных форм состоят в следующем:

1. каждая следующая нормальная форма в некотором смысле лучше предыдущей нормальной формы;
2. при переходе к следующей нормальной форме свойства предыдущих нормальных форм сохраняются.
3. В основе процесса проектирования лежит метод нормализации, т. е. декомпозиции отношения, находящегося в предыдущей нормальной форме, на

два или более отношений, которые удовлетворяют требованиям следующей нормальной формы

Итак, основные свойства отношений n-ой нормальной формы.

Первая нормальная форма

Отношение находится в 1НФ, если все его атрибуты являются простыми, все используемые домены должны содержать только скалярные значения. Не должно быть повторений строк в таблице.

Вторая нормальная форма

Отношение находится во 2НФ, если оно находится в 1НФ и каждый не ключевой атрибут неприводимо зависит от Первичного Ключа(ПК). Неприводимость означает, что в составе потенциального ключа отсутствует меньшее подмножество атрибутов, от которого можно также вывести данную функциональную зависимость.

Третья нормальная форма

Отношение находится в 3НФ, когда находится во 2НФ и каждый не ключевой атрибут нетранзитивно зависит от первичного ключа. Проще говоря, второе правило требует выносить все не ключевые поля, содержимое которых может относиться к нескольким записям таблицы в отдельные таблицы.

Нормальная форма Бойса-Кодда (НФБК) (частная форма третьей нормальной формы)

Определение 3НФ не совсем подходит для следующих отношений:

- 1) отношение имеет два или более потенциальных ключа;
 - 2) два и более потенциальных ключа являются составными;
 - 3) они пересекаются, т.е. имеют хотя бы один общий атрибут.
- Для отношений, имеющих один потенциальный ключ (первичный), НФБК является 3НФ.

Отношение находится в НФБК, когда каждая нетривиальная и неприводимая слева функциональная зависимость обладает потенциальным ключом в качестве детерминанта.

Четвертая нормальная форма

Отношение находится в 4НФ, если оно находится в НФБК и все нетривиальные многозначные зависимости фактически являются функциональными зависимостями от ее потенциальных ключей.

Пятая нормальная форма

Отношения находятся в 5НФ, если оно находится в 4НФ и отсутствуют сложные зависимые соединения между атрибутами.

Следующим шагом в создании базы данных является ограничение целостности данных.

Ограничение целостности — это некоторое утверждение, которое может быть истинным или ложным в зависимости от состояния базы данных.

База данных находится в согласованном (целостном) состоянии, если выполнены (удовлетворены) все ограничения целостности, определенные для базы данных.

Любое ограничение целостности является семантическим понятием, т.е. появляется как следствие определенных свойств объектов предметной области и/или их взаимосвязей.

Вместе с понятием целостности базы данных возникает понятие реакции системы на попытку нарушения целостности.

Классификация ограничений целостности

Ограничения целостности можно классифицировать несколькими способами:

- По способам реализации.

- По времени проверки.
- По области действия.

Классификация ограничений целостности по способам реализации

- Декларативная поддержка ограничений целостности.
- Процедурная поддержка ограничений целостности.

Классификация ограничений целостности по времени проверки

По времени проверки ограничения делятся на:

- Немедленно проверяемые ограничения.
- Ограничения с отложенной проверкой.

Классификация ограничений целостности по области действия

По области действия ограничения делятся на:

- Ограничения домена
- Ограничения атрибута
- Ограничения кортежа
- Ограничения отношения
- Ограничения базы данных

Ограничения целостности отношения представляют ограничения, накладываемые только на допустимые значения отдельного отношения, и не являющиеся ограничением целостности кортежа. Требование, что ограничение относится к отдельному отношению, означает, что для его проверки не требуется информации о других отношениях (в том числе не требуется ссылок по внешнему ключу на кортежи этого же отношения).

Первичный ключ (primary key)

Используется для обеспечения уникальности данных в столбцах и, в основном, для обеспечения ссылок на другие таблицы посредством связывания их внешними ключами.

Внешний ключ (foreign key)

Применяется вместе с определённым ранее первичным ключом или же ограничением уникальности (unique) в связанной таблице. Условие на значение внешнего ключа одной таблицы ставит в соответствие один или несколько столбцов другой таблицы.

Ограничение уникальности (unique)

Назначается чтобы запретить повторение значений в столбце таблицы. Для столбца, на котором определено ограничение первичного ключа, не может быть определено ограничение уникальности, так как уникальный индекс данного столбца уже создан.

Проверочное ограничение (check)

Устанавливает, какие значения может хранить столбец. Это ограничение, например, можно использовать для столбца, хранящего номера квартир в многоквартирном доме.

NotNull

Устанавливает поля, которые не могут быть пустыми.

Ссылочная целостность

Ссылочная целостность — это набор правил, которые определяют, что будет происходить при изменении значения родительского ключа, на который ссылается некоторый внешний ключ. Эти действия можно задавать независимо для операций обновления (ON UPDATE) или для операций

удаления (ON DELETE) записей в родительском отношении. Стандартом SQL определяется 4 типа действий, исполняемых по ссылке:

- **CASCADE.** Изменения значения родительского ключа автоматически приводят к таким же изменениям связанного с ним значения внешнего ключа. Удаление кортежа в родительском отношении приводит к удалению связанных с ним кортежей в дочернем отношении.
- **SET NULL.** Все внешние ключи, которые ссылаются на обновленный или удаленный родительский ключ получают значения NULL.
- **SET DEFAULT.** Все внешние ключи, которые ссылаются на обновленный или удаленный родительский ключ получают значения, принятые по умолчанию для этих ключей.
- **NO ACTION.** Значения внешнего ключа не изменяются. Если операция приводит к нарушению ссылочной целостности (появляются "висящие" ссылки), то такая операция не выполняется.

Дополнительные ограничения создаются при помощи так называемых триггеров.

Триггер — это особая разновидность хранимых процедур в базе данных. Особенность триггеров заключается в том, что SQL код, написанный в теле триггера, будет исполнен после того, как в базе данных произойдет какое-либо событие.

Главная особенность триггеров в том, что они позволяют работать с добавлением/изменением/удалением данных не только в рамках одной таблицы.

В PL/pgSQL создаются триггерные процедуры, которые вызываются при изменениях данных или событиях в базе данных. Триггерная процедура создаётся командой CREATE FUNCTION, при этом у функции не должно

быть аргументов, а типом возвращаемого значения должен быть trigger (для триггеров, срабатывающих при изменениях данных) или event_trigger (для триггеров, срабатывающих при событиях в базе). Для триггеров автоматически определяются специальные локальные переменные с именами вида PG_переменная, описывающие условие, повлекшее вызов триггера.

Последний этап — это тестирование базы данных. Для этого необходимо провести ряд разнообразных запросов к базе данных, по результатам которых мы выявим ошибки и исправим их, если такие имеются.

Запросы — это объект базы данных, который служит для извлечения данных из таблиц и предоставления их пользователю в удобном виде. В PostgreSQL оператором запроса является оператор select.

Глава 2. Практическая часть.

2.1 Описание предметной области.

Для разработки базы данных нужно разобраться с той сферой, для которой БД предназначена. Мы создаём базу данных для торгового центра, в котором сдаются в аренду торговые площади. Задачей является отслеживание финансовой стороны работы компании.

В базе данных фигурируют:

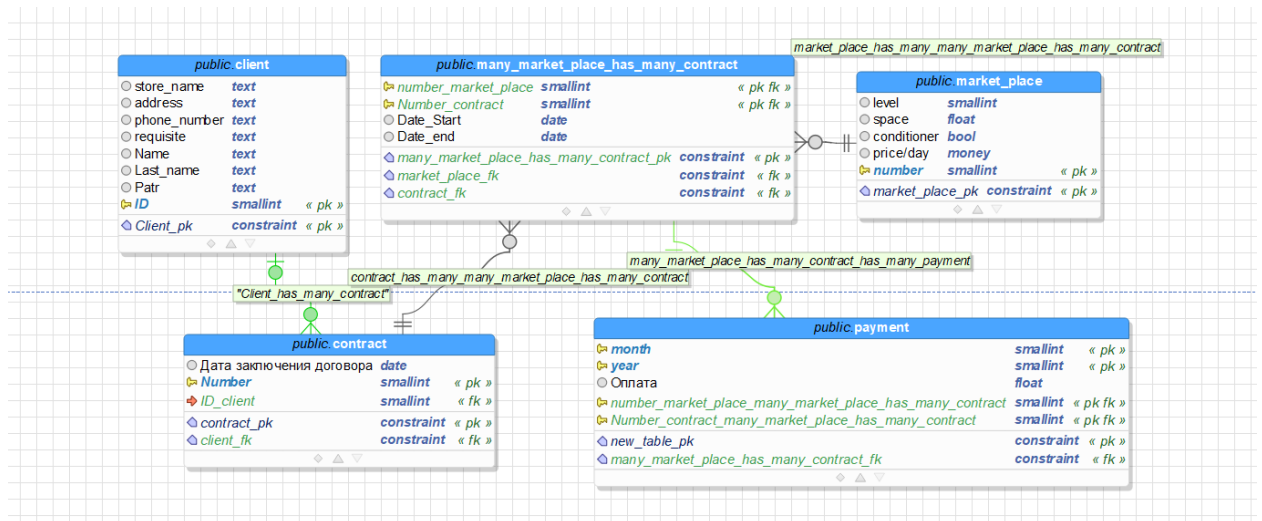
- **Торговая площадь**, характеризуется индивидуальным номером, площадью, этажом, наличием кондиционера, суточной ценой аренды.
- **Клиент**, характеризуется индивидуальным номером, название магазина, адрес, телефонный номер, банковские реквизиты, фамилия, имя, отчество.
- **Договор**, характеризуется датой заключения, номером и индивидуальным номером клиента. Клиент может заключать несколько договоров.

При заключении договора, указывается сроки аренда (Дата начала аренды и дата конца). Один договор может заключаться на несколько торговых площадей. Так же после заключения договора, должен вестись учёт платежей за аренду помещений.

На основе этих данных мы можем перейти к созданию схемы.

2.2 Диаграмма «Сущность-связь».

В процессе анализа рабочей сферы была создана следующая схема:



- 1) Сущность **Клиент** имеет 8 атрибутов (Название магазина, адрес, телефонный номер, банковские реквизиты, ФИО клиента и ID (первичный ключ)).
- 2) Сущность **Торговая площадь** имеет 5 атрибутов (Этаж, площадь, наличие или отсутствие кондиционера, цена за день и номер торговой площади (первичный ключ)).
- 3) Сущность **Договор** имеет 3 атрибута (Дата заключения договора, номер договора (первичный ключ) и ID клиента (внешний ключ)).
- 4) Сущности 2 и 3 связаны (M:M), но т.к. клиент по одному договору может арендовать несколько помещений с разной продолжительностью аренды, поэтому таблица этой связи имеет 4 атрибута (Дата начала и конца аренды и 2 foreign key и у них ссылочная целостность cascade).
- 5) Сущность **Оплата** имеет 5 атрибутов (Месяц (первичный ключ), год (первичный ключ), оплата, номер торговой площади (pk fk) и номер договора (pk fk)).

В данной диаграмме мы наглядно видим каждый элемент нашей базы данных и его связь с другими элементами. Используя данную схему мы не только визуализируем работу нашей базы но и создаем ее реализацию. Она создается автоматически на языке SQL.

2.3 Нормализация реляционной модели. Нормальные формы.

Чтобы заняться нормализацией данных, сначала нам нужно собрать в одну таблицу все известные нам атрибуты и задать пару примеров данных.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Клиент						Договор				Помещение					оплата		
id клиента	Название	Адресс	Тел. Номер	Реквизиты	Ф.И.О	id договора	Дата заключения договора	Дата начала	Дата конца	Номер тоговой площади (id)	Этаж	Площадь(М²З)	Кондиционер	Цена/день	оплата	год	месяц
1	-	-	-	-	-	1	08.09.2001	11.09.2001	20.11.2005	1	-	-	-	500	2000	2001	11
1	-	-	-	-	-	2	08.07.2002	11.08.2001	20.01.2004	5	-	-	-	500	3000	2002	1
1	-	-	-	-	-	4	08.03.2001	21.03.2001	24.02.2006	2	-	-	-	600	-	-	-
2	-	-	-	-	-	3	02.03.2002	04.04.2002	06.04.2007	3	-	-	-	400	-	-	-
1	-	-	-	-	-	1	08.09.2001	11.09.2001	20.11.2005	1	-	-	-	500	3000	2002	4
1	-	-	-	-	-	2	08.09.2001	03.07.2003	20.11.2005	4	-	-	-	500	3000	2002	1

Таблица 1

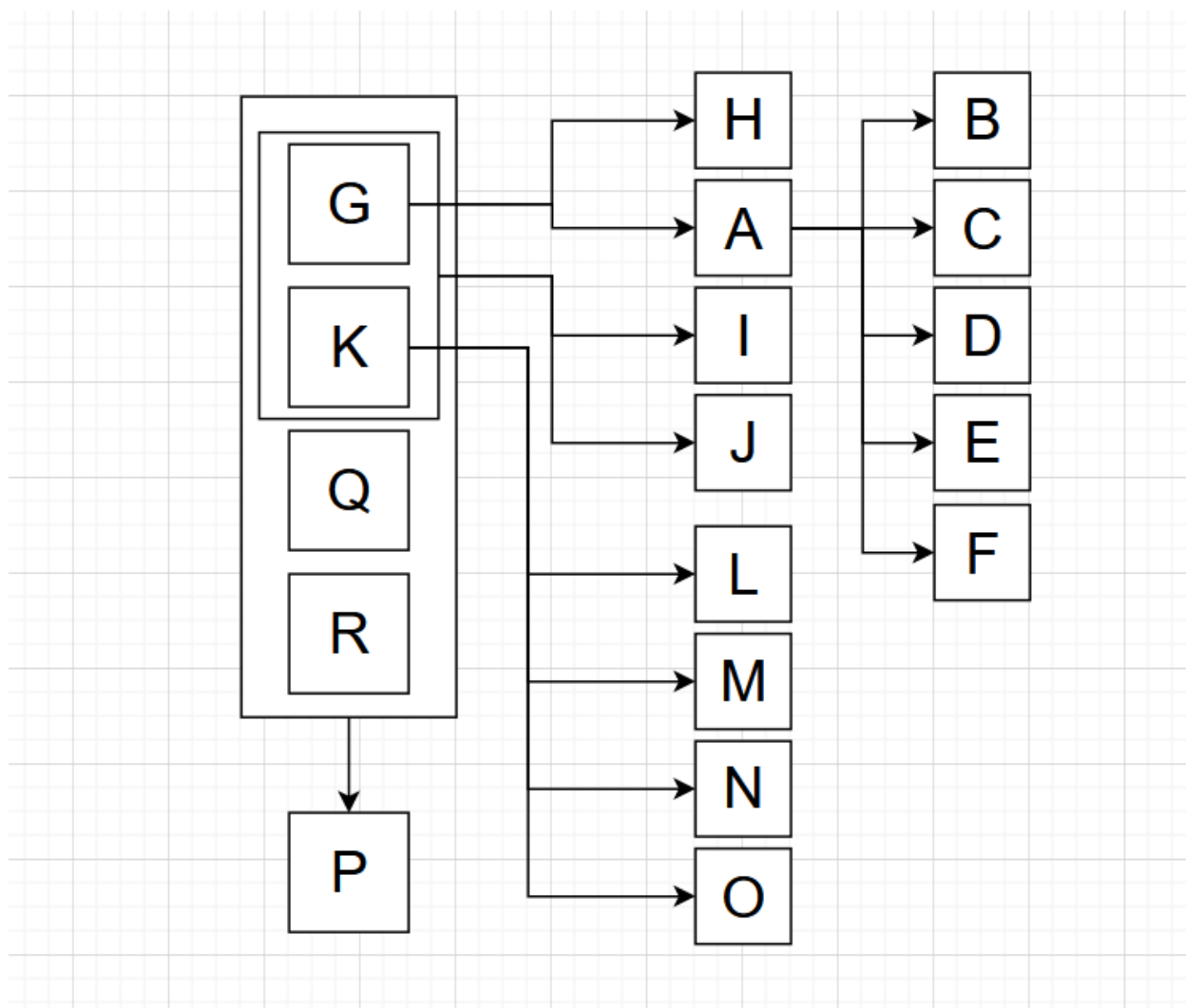
Нетрудно заметить, что из-за многозначности (Оплата, год, месяц) данные приходится дублировать, чтобы не потерять некоторую информацию.

Primary Key(G, K, Q, R) – получившийся ключ данной базы данных.

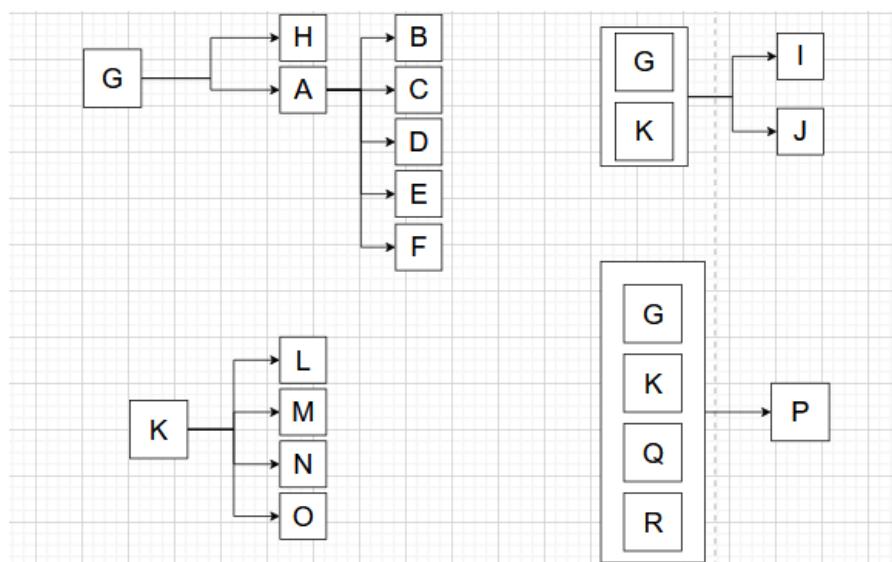
Обратим внимание на имеющиеся проблемы(аномалии):

- Не можем добавить клиента не добавляя договор.
- Удаляя договор удалим клиента.
- Удаляя оплату удаляем помещение.

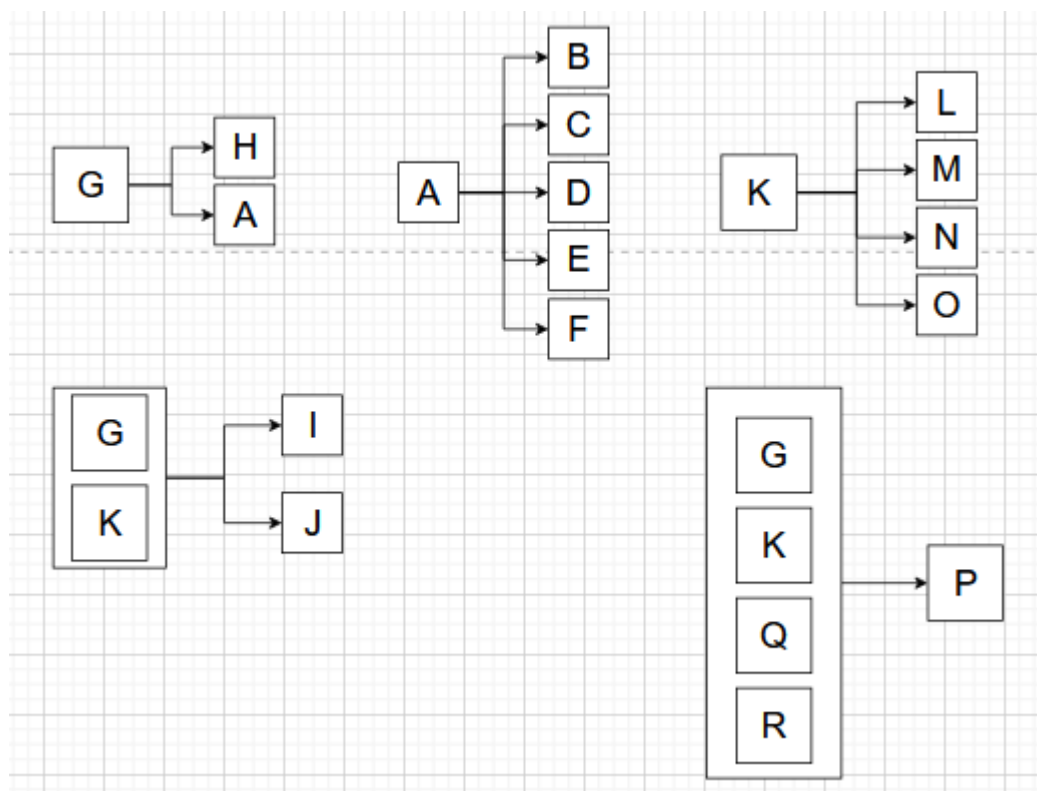
Функциональные зависимости (Диаграммы функциональных зависимостей).



2-я нормальная форма:



3-я нормальная форма:



2.4 Описание декларативных ограничений целостности.

Таблица	Атрибут	Вид ограничения	Описание	скрипт
Клиент (client)	id клиента	первичный ключ	id клиента должен быть уникальным	CONSTRAINT "Client_pk" PRIMARY KEY ("ID")
Договор (contract)	Номер договора	первичный ключ	Номер договора должен быть уникальным	CONSTRAINT contract_pk PRIMARY KEY ("Number")
Торговая площадь (market_place)	Номер торговой площади	первичный ключ	Номер площади должен быть уникальным	CONSTRAINT market_place_pk PRIMARY KEY (number)
many_market_place_has_many_contract	Номер торговой площади	Внешний и первичный ключ	Контракт должен иметь уникальный номер	CONSTRAINT many_market_place_has_many_contract_pk PRIMARY KEY ("Number_contract")
Оплата (payment)	Месяц, год	первичный ключ	Оплата происходит за определённый месяц	CONSTRAINT new_table_pk PRIMARY KEY (month,year,number_market_place_has_many_contract,"Number_contract_many_market_place_has_many_contract")

2.5 Реализация запросов.







Базу данных можно протестировать несколькими запросами, представленными ниже:

1) Определить список занятых ТП на заданную дату.

Реализация на SQL:

```
1 with Aboba as
2 (select * from "market_place" as A
3 join "many_market_place_has_many_contract" as B
4 ON (A."number" = B."number_market_place")
5 where current_date between B."Date_Start"
6 and B."Date_end")
7 select "number", "level", "Number_Contract", "Date_Start", "Date_end" from aboba;
```

Результат:







Результат		План выполнения		Сообщения		Notifications	
	number smallint	 level smallint	 Number_Contract smallint	 Date_Start date	 Date_end date		
1	1	1		1	2018-03-13	2026-01-01	
2	3	1		3	2019-02-01	2023-11-20	
3	4	1		4	2019-09-08	2027-02-11	
4	5	1		5	2021-03-08	2023-09-06	
5	7	3		6	2021-06-15	2030-11-02	

2) Определить есть ли у человека долги либо переплаты.

Реализация на SQL:

```
1 with aboba as
2 (select cl."ID" as "id магазина", "store_name" as "Магазин", "Number" as "контракт", SE."Date_Start",
3 SE."Date_end", pay."Оплата", mp."price/day"
4 from "Client" as cl
5 join "Contract" as con on (cl."ID" = con."ID_Client")
6 join "many_market_place_has_many_contract" as SE on (se."Number_Contract" = con."Number")
7 join "Payment" as pay on (pay."Number_Contract_many_market_place_has_many_contract" = con."Number")
8 join "market_place" as mp on (mp."number" = SE."number_market_place"))
9
10 select "контракт", (current_date - "Date_Start") as "Кол-во дней",
11 (((current_date - "Date_Start") * "price/day")::numeric as "Должно быть уплачено",
12 SUM("Оплата") as "Всего уплачено",
13 (((current_date - "Date_Start") * "price/day")::numeric - SUM("Оплата"))::numeric as "Долг"
14 from aboba
15 group by "контракт", "Date_Start", "price/day" ;
```

Результат:

Результат		План выполнения		Сообщения	Notifications					
	контракт smallint		Кол-во дней integer		Должно быть уплачено numeric		Всего уплачено double precision		Долг numeric	
1		1		1197		598500.00		96000		502500
2		2		7132		3566000.00		54000		3512000
3		2		1109		665400.00		54000		611400
4		3		872		479600.00		33000		446600
5		5		106		47700.00		55500		-7800

3) Три торговые точки, которые находятся в аренде дольше всего.

Реализация на SQL:

```
1 with Aboba as
2 (select * from "market_place" as A
3 join "many_market_place_has_many_contract" as B
4 ON (A."number" = B."number_market_place")
5 where current_date between B."Date_Start"
6 and B."Date_end")
7 select "number", "level", "Number_Contract", "Date_Start", "Date_end",
8 (current_date - "Date_Start") as "Кол-во дней" from aboba
9 order by "Кол-во дней" desc
10 limit 3;
```

Результат:

Результат		План выполнения		Сообщения	Notifications		
<div><div></div></div> number smallint	<div><div></div></div> level smallint	<div><div></div></div> Number_Contract smallint	<div><div></div></div> Date_Start date	<div><div></div></div> Date_end date	<div><div></div></div> Кол-во дней integer		
1	1	1	1	2018-03-13	2026-01-01		
2	3	1	3	2019-02-01	2023-11-20		
3	4	1	4	2019-09-08	2027-02-11		

4) Найти договоры, в которых все ТП арендуются одновременно.

Реализация на SQL:

```
1 with aboba as (select con."Number" as "Номер контракта", SE."number_market_place",
2                     SE."Date_Start", SE."Date_end"
3 from "Contract" as con
4 join "many_market_place_has_many_contract" as SE on (SE."Number_Contract" = con."Number"))
5 select * from aboba
6 where (Select count(*) from aboba as aboba2 where
7        aboba."Date_Start" = aboba2."Date_Start") > 1;
```

Результат:

	Номер контракта smallint	number_market_place smallint	Date_Start date	Date_end date
1	2	1	2001-12-12	2002-11-11
2	2	2	2001-12-12	2002-11-11

5) Прибыль предприятия за год.

Реализация на SQL:

```
1 select SUM("Оплата") as "Прибыль за год" from "Payment"
2 where "Год" = 2019;
```

Результат:

	Прибыль за год double precision
1	33000

2.6 Реализация процедурных ограничений целостности.

Перед реализацией триггера, нужно понять, что он должен делать и для чего он создаётся.

Если торговая площадь находится в аренде, то её нельзя добавлять в новый договор.

Посмотрев на схему, понимаем, что данный триггер должен срабатывать, замещая команду `before insert` “many_market_place_has_many_contract”.

То есть в тот момент, когда мы пытаемся добавить кортеж в таблицу many_market_place_has_many_contract.

Реализация на SQL:

```
CREATE OR REPLACE FUNCTION GetTable(n integer, a date, b date)
```

```
returns table (num smallint)
```

```
AS
```

```
$$
```

```
BEGIN
```

```
return query (
```

```
select number_market_place from many_market_place_has_many_contract
```

```
where (number_market_place = n) and (("Date_Start" <= a) and (a <= "Date_end"))
```

```
or (("Date_Start" <= b) and (b <= "Date_end")));
```

```
END;
```

```
$$
```

```
LANGUAGE 'plpgsql';
```

```
create or replace function trigger_1()
```

```
returns trigger AS
```

```
$$
```

BEGIN

IF (select count(*) from GetTable(NEW."number_market_place",
NEW."Date_Start", NEW."Date_end")) >0

THEN RAISE EXCEPTION 'Ошибка ввода'

USING HINT = 'Нельзя указывать занятую торговую площадь';

END IF;

RETURN NEW;

END;

\$\$

LANGUAGE 'plpgsql';

create trigger t1

after insert on many_market_place_has_many_contract

FOR EACH ROW

EXECUTE PROCEDURE trigger_1();

Пример использования:

Пытаемся связать договор с занятой, на данный момент, торговой площадью.

	number_market_place [PK] smallint		Date_Start date		Date_end date		Number_Contract [PK] smallint	
4		2	2022-02-12		2023-02-12		6	
5		3	2019-02-01		2023-11-20		3	
6		4	2019-09-08		2027-02-11		4	
7		4	2018-09-09		2019-07-07		6	
8		5	2021-03-08		2023-09-06		5	
9		7	2021-06-15		2030-11-02		6	
10		7	08.09.2025		08.09.2030		1	

Торговая площадь под номером 7 уже арендуется и при попытке её присвоить другому договору, мы увидим срабатывание триггера.

	number_market_place [PK] smallint		Date_Start date		Date_end date		Number_Contract [PK] smallint	
4		2	2022-02-12		2023-02-12		6	
5		3	2019-02-01		2023-11-20		3	
6		4	2019-09-08		2027-02-11		4	
7		4	2018-09-09		2019-07-07		6	
8		5	2021-03-08		2023-09-06			
9		7	2021-06-15		2030-11-02			
10		7	08.09.2025		08.09.2030			

ОШИБКА: Ошибка ввода HINT: Нельзя указывать занятую торговую площадь CONTEXT: функция PL/pgSQL trigger_1(), строка 4, оператор RAISE .

Заключение.

Результатом выполнения курсовой работы стало разработанное приложение баз данных, позволяющее автоматизировать операции учета и управления торговыми площадями в торговом центре. Разработанное приложение отвечает всем требованиям предметной области, таблицы созданной базы данных отвечают требованиям нормализации, что позволяет обеспечить целостность и непротиворечивость информации.

Приложение позволяет решать все задачи, сформулированные в задании на курсовую работу. Это позволяет сделать вывод о том, что задание выполнено полностью.

Разработанная в курсовой база данных легко дополняется при необходимости разработки профессиональной базы данных.

Также в ходе выполнения практической части были приобретены важные навыки работы с базами данных:

- Умение создавать запросы.
- Умение смоделировать и нормализовать базу данных, так как, понимая ее архитектуру с нуля, можно сделать все что угодно, даже изначально не понимая, как это работает.
- Всегда обращать внимание на структурную целостность любой информации.

Список используемых источников

1. Марков А.С. Базы данных. Введение в теорию и методологию: учебник / А.С. Марков, К.Ю. Лисовкий. – М.: Финансы и статистика, 2004. – 512 с. – ISBN 5-279-02298-5.
2. Habr [Электронный ресурс]/ Режим доступа: <https://habr.com/ru/post/254773/>
3. Wikiedia [Электронный ресурс]/ Режим доступа: <https://ru.wikipedia.org>
4. Руководство по PostgreSQL [Электронный ресурс]/ Режим доступа: <https://metanit.com/sql/postgresql/>
5. Петров В.Н. Информационные системы / В.Н. Петров. – СПб.: Питер, 2002. – 688 с. – ISBN 5-318-00561-6.
6. Риккарди Г. Системы баз данных. Теория и практика использования в Internet и среде Java. / Грег Риккарди; пер. с англ. – М.: Издательский дом «Вильямс», 2001. – 480 с. – ISBN 5-8459-0208-8 (рус.).
7. Саак А.Э. Информационные технологии управления: учебник для вузов / А.Э. Саак, Е.В. Пахомов, В.Н. Тюшняков. – СПб.: Питер, 2005. – 320 с. ISBN 5-469-00412-0.