Оглавление

Введение

Одним из важных потребностей в жизни человека является отдых. От качества отдыха зависит эффективность человека в его жизни. Человеческому организму требуется не только отдых в физическом плане, но и в моральном тоже. Что иное, если не игры, помогают современному человеку отдохнуть от повседневной рутины?

Развитее игровой индустрии в современном мире находится на высоком уровне. Многообразие игр с их захватывающим сюжетом способствуют привлечению всё большему числа игроков. Сами игры, выйдя в сеть, становятся не просто сферой отдыха и развлечения, но вырастает в отдельную ветку киберспорта.

Казалось бы, что сложного в этих играх? Но разработка больших игр занимает отнюдь не пару дней, как кто-нибудь мог бы предположить. Одной из важных задач в проектировании игры это создание базы данных.

Ставятся следующие задачи:

- Описание предметной области
- проектирование базы данных,
- разработка базы данных,
- интеграция базы данных в игру
- поддержка и внедрение базы данных.

1. Основная часть

1.2. Исследование предметной области

Игровая индустрия разделятся на два лагеря. Это ААА игры и инди игры. Это как две стороны медали, которые вместе образуют индустрию. Но хоть ААА и инди игры образуют одну область. Они живут обособленно друг от друга, и имеют массу отличий. У них есть одна общая черта, в обоих случаях вся информация храниться в базах данных. В базах данных хранятся от никнеймов игроков до их личной статистики.

ААА проекты, разрабатываются огромными командами с отдельными штатами сотрудников, в одном таком штате могут числится от 100 человек разных специальностей. Если команда не справляется со сроками, к ним могут подключить дополнительные силы для реализации проекта в срок. Как правило, такие продукты выпускаются издателями, а не самими разработчиками.

Если говорить про инди игры, тут ситуация диаметрально противоположна. Такие продукты разрабатываются инди компаниями, как понятно из названия "Инди игры". Команда разработчиков в таких компаниях, как правило, на порядок меньше, бюджеты ниже, а может быть, что бюджета вовсе нет, и всё создание продукта держится на энтузиазме. Конечно, инди игры разрабатывают не только командами разработчиков, но также и отдельными личностями. Выпускаются на рынок такие игры под своим издательством. После выхода в Store, команду разработчиков ждёт три пути. Первый - это полный провал в продажах и как следствие, раскол команды. Так как работа на энтузиазме долго держаться не может. Разработчики уходят в другие более крупные проекты. Проекты, которые застали второй путь, о котором написано далее. Второй путь - игра привлекает пользователей, тем самым расширяя возможности команды разработчиков. После, разработчики приступают к новому проекту, с расширенными бюджетами и некой известностью среди геймерского сообщества. Эта известность, безусловно, увеличит шансы на успех будущего проекта. Третий - игру также постиг успех.

Но командой заинтересовался крупный игрок на рынке. После чего разработчиков покупают либо образуют новый штат сотрудников или распределяют в уже существующие.

Но всё что было сказано в предыдущем абзаце, актуально лишь для Desktop, Pastgen и Nextgen. Ведь на рынке мобильных игр ситуация обстоит иначе. Множество игр на мобильных устройствах разрабатываются одним проектировщикам, так-как ресурсы создания таких игр на порядок меньше, чем на других устройствах. Не требуют таких строгих сроков, усилий, бюджета. Потому что игроки на мобильных устройствах не требуют глубину сюжета, разнообразие игровых механик, проработанности мира, передовой графики и т.д. Для игры не телефоне достаточно одной двух интересных механик и не графику уровня первого Doom.

Игра "i don't know" будет реализована на игровом движке Unity5 3D. Все скрипты будут написаны на языке C#. Для анимирования и моделирования будут использованы такие программы, как Blender 3D и Magic Voxel. Для текстурированния, Adobe Substance Painter 3D. В качестве IDE будет использован Rider.

Сюжет игры будет разворачиваться в Канаде, район в 100 километрах от поселения Бейкер-Лейк, Нунавут. Во время суровой зимы. Вы как геолог, попадаете в эти непростые условия. Изначальной задачей вашей экспедиции был Голубой гранат, цена которого составляет более чем 2 млн. долларов. Но внезапная смена погоды прерывает ваши планы. Вертолёт, на котором вы летели потерпел крушение, и вся группа гибнет, кроме вас. Единственной вашей целью после этого является выживание. Это будет не самой лёгкой задачей, ведь район где было совершенно крушение заброшен. Высокие холода заставили жителей этих мест покинуть свои дома.

1.3. Создание модели базы данных

В основе моделирования базы данных лежит идея проектирования структуры базы данных, которая определяет, как можно получить доступ к

хранимой информации, категоризировать ее и манипулировать ею. Это основа проектирования базы данных, а используемая конкретная модель данных определяет схему базы данных и общие усилия по разработке. База данных должна быть простой в использовании и должна поддерживать целостность данных безопасным образом. Надежная модель базы данных также позволит использовать различные способы управления, контроля и организации хранимой информации для эффективного выполнения нескольких ключевых задач. На этапе проектирования схемы базы данных будут предоставлять необходимую документацию по каналам передачи данных, что упрощает функциональность базы данных.

Типы методов моделирования баз данных.

Ниже приведен список наиболее распространенных методов моделирования баз данных. Нужно обратить внимание, что в зависимости от типа данных и потребностей конечного пользователя при доступе к базе данных можно использовать несколько моделей для создания более сложной структуры базы данных. Конечно, в любом случае для установления и поддержания высоких операционных стандартов потребуется создание диаграмм базы данных. К счастью, готовые инструменты для создания диаграмм и дизайна, такие как Creately, могут упростить эту задачу. Из перечисленных ниже моделей реляционная модель является наиболее часто используемой моделью для большинства проектов баз данных. Но в некоторых особых случаях другие модели могут оказаться более полезными.

- **Реляционная модель** (рис. 1). Основанная на математической теории, эта модель базы данных выводит хранение и поиск информации на новый уровень, поскольку предлагает способ находить и понимать различные отношения между данными. Глядя на то, как различные переменные могут изменить отношения между данными, можно получить новые перспективы, поскольку представление информации изменяется путем сосредоточения внимания на различных атрибутах или областях. Эти модели часто можно найти в системах бронирования авиабилетов или в банковских базах

данных.

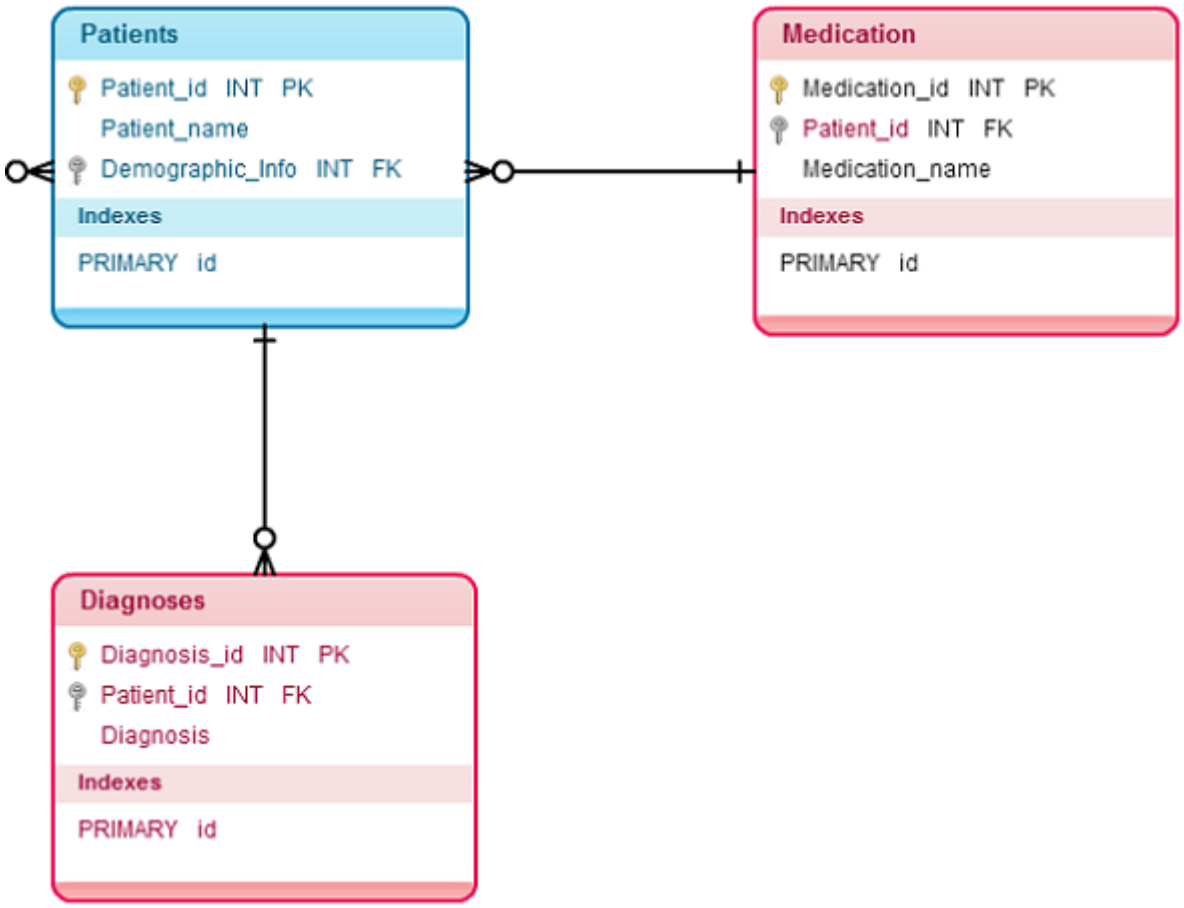

Рисунок 1 - Реляционная модель.

– **Графическая модель**. Графическая модель — еще одна набирающая популярность модель. Эти базы данных созданы на основе теории графов и используют узлы и ребра для представления данных. Структура чем-то похожа на объектно-ориентированные приложения. Базы данных графов, как правило, легче масштабировать, и они обычно работают быстрее для ассоциативных наборов данных.

– **Иерархическая модель** (рис. 2). Подобно обычной организационной схеме, используемой для организации компаний, эта модель базы данных имеет такой же древовидный вид и часто используется для структурирования XML-документов. С точки зрения эффективности данных это идеальная модель, в которой данные содержат вложенную и отсортированную информацию, но она может быть неэффективной, если данные не имеют восходящей связи с основной точкой данных или предметом. Эта модель

хорошо работает для системы управления информацией о сотрудниках в компании, которая стремится ограничить или назначить использование оборудования определенным лицам и/или отделам.

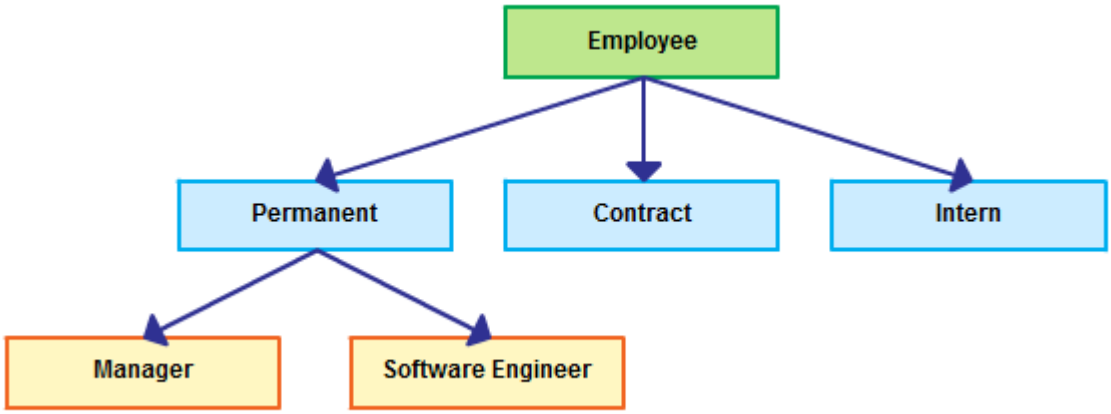

Рисунок 2 - Иерархический метод, самая первая модель проектирования баз данных.

– **Сетевая модель** (рис. 3). Используя записи и наборы, эта модель использует подход отношения «один ко многим» для записей данных. Несколько ветвей выделяются для структур более низкого уровня и ветвей, которые затем соединяются несколькими узлами, которые представляют структуры более высокого уровня в рамках информации. Этот метод моделирования базы данных обеспечивает эффективный способ извлечения информации и организации данных, чтобы их можно было рассматривать несколькими способами, предоставляя средства повышения эффективности бизнеса и времени реакции. Это жизнеспособная модель для планирования дорог, поездов или инженерных сетей.

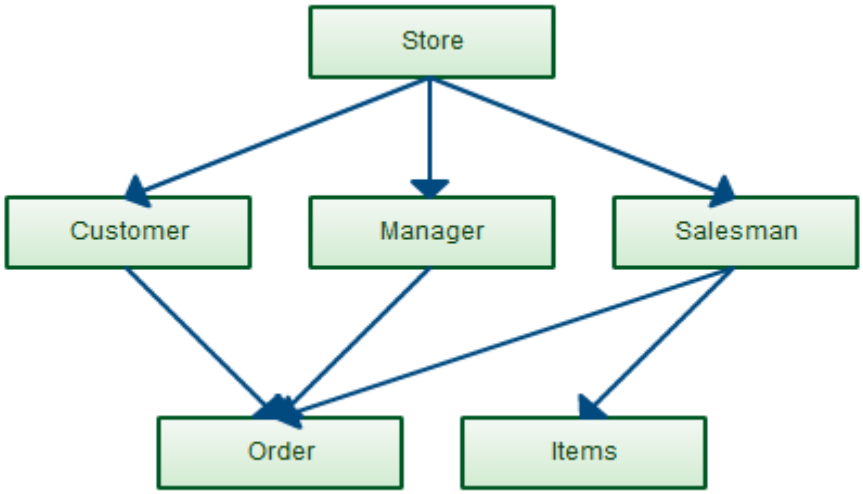

Рисунок 3 - Сетевая модель, в которой узел может иметь несколько родительских узлов.

– **Многомерная модель**: это адаптация реляционной модели, которая часто используется в сочетании с ней путем добавления «измерения» факта к точкам данных. Эти факты можно использовать в качестве мерила для других данных, чтобы определить, как размер группы или время группы повлияли на определенные результаты. Это может помочь бизнесу принимать более эффективные стратегические решения и помочь узнать свою целевую аудиторию. Эти модели могут быть полезны организациям, занимающимся анализом продаж и прибыли.

– **Объектно-реляционная модель**: эти модели создали базу данных совершенно нового типа, которая сочетает в себе дизайн базы данных с прикладной программой для решения конкретных технических проблем, используя лучшее из обоих миров. На сегодняшний день объектные базы данных все еще нуждаются в доработке для достижения большей стандартизации. Реальные приложения этой модели часто включают технические или научные области, такие как инженерия и молекулярная биология.

Из всех вышеперечисленных моделей баз данных, больше всего для этого проекта подходить реляционная модель. Основным преимуществом подхода реляционной базы данных является возможность создания значимой информации путем объединения таблиц. Объединение таблиц позволяет понять отношения между данными или то, как таблицы соединяются. SQL включает в себя возможность подсчитывать, добавлять, группировать, а также комбинировать запросы. SQL может выполнять основные математические функции и промежуточные итоги, а также логические преобразования. Аналитики могут упорядочить результаты по дате, имени или любому столбцу.

Реляционные базы данных имеют ряд преимуществ по сравнению с другими форматами баз данных:

Гибкость

SQL имеет встроенный язык для создания таблиц, называемый языком определения данных (DDL). DDL позволяет добавлять новые столбцы, новые таблицы, переименовывать отношения и вносить другие изменения даже во время работы базы данных и выполнения запросов.

Уменьшенная избыточность

Реляционные базы данных устраняют избыточность данных. Информация для одного клиента отображается в одном месте — в одной записи в таблице клиентов. В таблице заказов нужно хранить только ссылку на таблицу клиентов. Практика разделения данных во избежание избыточности называется нормализацией. Разработчики прогрессивных баз данных следят за тем, чтобы таблицы нормализовались в процессе проектирования.

Простота резервного копирования и аварийного восстановления

Реляционные базы данных являются транзакционными — они гарантируют согласованность состояния всей системы в любой момент. Большинство реляционных баз данных предлагают простые варианты экспорта и импорта, что упрощает резервное копирование и восстановление. Этот экспорт может происходить даже во время работы базы данных, что упрощает восстановление в случае сбоя. Современные облачные реляционные базы данных могут выполнять непрерывное зеркальное отображение, благодаря чему потеря данных при восстановлении измеряется секундами или даже меньше. Большинство облачных сервисов позволяют создавать реплики чтения. Эти реплики чтения позволяют хранить копию данных только для чтения в облачном центре обработки данных. Реплики также можно повысить до экземпляров с возможностью чтения и записи для аварийного восстановления.

Но без нормализации в реляционных базах данных не обойтись.

Нормализация базы данных - это процесс, используемый для организации базы данных в таблицы и столбцы. Существует шесть основные форм: первая нормальная форма, вторая нормальная форма и так далее до шестой. Основная

идея заключается в том, что каждая таблица должна быть посвящена определенной теме и включать только вспомогательные темы. Избыточные данные тратят дисковое пространство и создают проблемы с обслуживанием. Если необходимо изменить данные, существующие более чем в одном месте, данные должны быть изменены точно так же во всех местах. Изменение адреса клиента намного проще реализовать, если эти данные хранятся только в таблице Customers и нигде больше в базе данных. И именно поэтому необходимо нормализовать все таблицы в базе.

Изобретатель реляционной модели Эдгар Кодд предложил теорию нормализации данных с введением Первой нормальной формы, а сам продолжил расширять теорию со Второй и Третьей нормальной формой. Позже он присоединился к Раймонду Ф. Бойсу для разработки теории нормальной формы Бойса-Кодда.

Теория нормализации данных все еще развивается дальше. Например, есть обсуждения даже по 6-й нормальной форме. Однако в большинстве практических приложений нормализация достигает своего наилучшего результата в 3-й нормальной форме. Эволюция нормализации в теориях SQL (рис. 4) проиллюстрирована ниже:

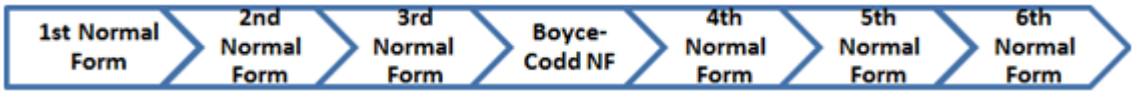

Рисунок 4 – База данных нормальных форм

Пример нормализации базы данных можно легко показать с помощью case study. Предположим, видеотека поддерживает базу данных фильмов, взятых в аренду. Без какой-либо нормализации в базе данных вся информация хранится в одной таблице, как показано ниже (рис. 5). Ниже приведена нормализация с решением:

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean, Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal, Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Рисунок 5 – Пример таблицы

Правила 1NF (первая нормальная форма)

- Каждая ячейка таблицы должна содержать одно значение.
- Каждая запись должна быть уникальной.

1NF Пример (рис. 6)

FULL NAMES	PHYSICAL ADDRESS	MOVIES RENTED	SALUTATION
Janet Jones	First Street Plot No 4	Pirates of the Caribbean	Ms.
Janet Jones	First Street Plot No 4	Clash of the Titans	Ms.
Robert Phil	3rd Street 34	Forgetting Sarah Marshal	Mr.
Robert Phil	3rd Street 34	Daddy's Little Girls	Mr.
Robert Phil	5th Avenue	Clash of the Titans	Mr.

Рисунок 6 - Пример 1NF в СУБД

КЛЮЧ в SQL - это значение, используемое для уникальной идентификации записей в таблице. Ключ SQL - это один столбец или комбинация нескольких столбцов, используемых для уникальной идентификации строк или кортежей в таблице. Ключ SQL используется для выявления дубликатов информации, а также помогает установить связь между несколькими таблицами в базе данных.

Первичный - это значение одного столбца, используемое для уникальной идентификации записи базы данных.

Он имеет следующие атрибуты

- Первичный ключ не может быть нулевым
- Значение первичного ключа должно быть уникальным
- Значения первичного ключа следует менять редко
- Первичному ключу должно быть присвоено значение при вставке новой записи.

Составной ключ (рис. 7) - это первичный ключ, состоящий из нескольких столбцов, используемых для уникальной идентификации записи.

В этой базе данных есть два человека с одинаковым именем Роберт Фил, но они живут в разных местах.

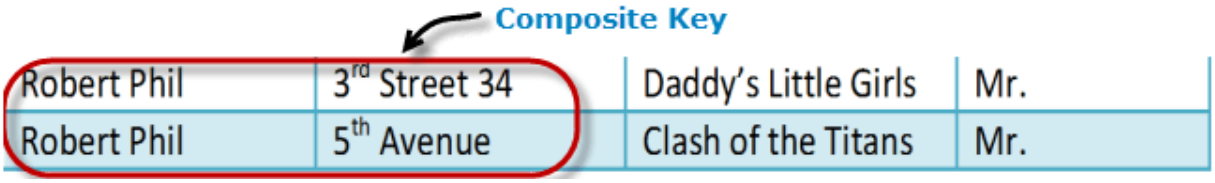

Рисунок 7 – Составной ключ в базе данных

Правила 2NF (вторая нормальная форма)

Правило 1- Быть в 1NF.

Правило 2 - первичный ключ с одним столбцом, который функционально не зависит от какого-либо подмножества отношения ключа-кандидата.

Правило 3 - Никакие непустые атрибуты функционально не зависят от подмножества ключей-кандидатов. Другими словами, любой столбец, который не является ключевым столбцом, зависит от всей информации в ключе-кандидате.

Неофициально вторая нормальная форма утверждает, что все атрибуты должны зависеть от всего ключа-кандидата.

Ключ-кандидат - это минимальный набор атрибутов, который определяет другие атрибуты, включенные в отношение. Это минимально в том, что если вы удалили один атрибут, остальные атрибуты не образуют ключ-кандидат.

Что это значит? Если значение атрибута A определяется значением атрибута S, то A ФУНКЦИОНАЛЬНО ЗАВИСИТ от S., например, ваш возраст функционально зависит от даты вашего рождения.

Ниже предоставлена таблица (рис. 8) примера отношений которое не удовлетворяет 2 NF:

часть	поставщик	Количество	страна-поставщик
Седло	Bikeraft	10	США
Тормозной рычаг	Tripebike	5	Италия
Верхняя трубка	UpBike	3	Канада
Седло	Tripebike	8	Италия

Рисунок 8 - Склад запчастей для велосипедов

– Ключ-кандидат - это набор деталей и поставщиков, который выражается следующим образом {part, supplier}.

– Непустыми атрибутами (которые не являются частью ключа-кандидата) являются количество и страна-поставщик.

– Существуют функциональные зависимости между частью, поставщиком и количеством (выраженные как часть, поставщик → количество) и между поставщиком и страной-поставщиком (выраженные как поставщик → страна-поставщик).

Почему это не удовлетворяет 2NF? Набор {part, supplier} является единственным ключом-кандидатом этого отношения. Стоимость страны-поставщика функционально зависит от поставщика. Страна-поставщик не является частью ключа-кандидата, поэтому она не является простым атрибутом и функционально зависит от части ключа-кандидата, а не от всего ключа-кандидата {part, supplier}.

Чтобы преобразовать это отношение в 2NF (рис. 9), нам нужно разделить его на два отношения: части велосипеда (с атрибутами part, supplieri quantity) и Поставщики (с атрибутами supplier и supplier country). Это будет выглядеть следующим образом:

часть	поставщик	Количество
Седло	Bikeraft	10
Тормозной рычаг	Tripebike	5
Верхняя трубка	UpBike	3
Седло	Tripebike	8

Рисунок 9 - Части велосипеда

Отношение частей находится в 2NF (рис. 10), потому что, как и раньше, атрибут количества зависит от пары поставщик и деталь.

поставщик	страна-поставщик
Bikeraft	США
Tripebike	Италия
UpBike	Канада

Рисунок 10 – Поставщики

Отношение поставщиков находится в 2NF, потому что страна-поставщик функционально зависит от поставщика, который является ключом-кандидатом этого отношения.

Правила 3NF (третья нормальная форма)

- Правило 1- Быть в 2NF
- Правило 2- Не имеет транзитивных функциональных зависимостей
- Все несложные атрибуты напрямую (нетранзитивно) зависят от всего ключа-кандидата.

Другими словами, несущественные атрибуты должны функционально зависеть от ключа (рис. 11), но они не должны зависеть от другого несущественного атрибута. Не простые атрибуты 3NF зависят от “ничего, кроме ключа”.

order_id	Дата	клиент	электронная почта клиента
1/2020	2020-01-15	Джейсон Уайт	white@example.com
2/2020	2020-01-16	Мэри Смит	msmith@mailinator.com
3/3030	2020-01-17	Джейкоб Альбертсон	jasobal@example.com
4/2020	2020-01-18	Боб Дикинсон	bob@fakemail.com

Рисунок 11 – Информация о порядке

- Ключ кандидата: order_id
- Не простые атрибуты: дата, клиент, электронная почта клиента
- Функциональные зависимости: date зависит от order_id (order_id → date); customer зависит от order_id (order_id → customer), a customer email зависит от customer (customer → customer email).

Это соотношение не удовлетворяет 3NF. Единственным ключом-кандидатом в этом отношении является order_id. Значение customer email функционально зависит от атрибута customer, который не является простым атрибутом. Таким образом, отношение нарушает 3NF.

Опять же, мы разделяем это на два отношения (рис. 12 - 13): заказы (с атрибутами order_id, dateи customer) и Клиенты (с атрибутами customer и customer email):

order_id	Дата	клиент
1/2020	2020-01-15	Джейсон Уайт
2/2020	2020-01-16	Мэри Смит
3/3030	2020-01-17	Джейкоб Альбертсон
4/2020	2020-01-18	Боб Дикинсон

Рисунок 12 – Заказы

клиент	электронная почта клиента
Джейсон Уайт	white@example.com
Мэри Смит	msmith@mailinator.com
Джейкоб Альбертсон	jasobal@example.com
Боб Дикинсон	bob@fakemail.com

Рисунок 13 – Клиенты

Orders находится в 3NF, потому что атрибуты date и customer не нарушают правило 3NF; их значения зависят от номера order_id . Клиенты находятся в 3NF, потому что электронная почта клиента функционально зависит от клиента, который является ключом-кандидатом этого отношения. В обоих случаях все несложные атрибуты зависят от ключа-кандидата.

Первая, вторая и третья нормальные формы являются основными нормальными формами в нормализации базы данных:

– Первая нормальная форма (1NF) утверждает, что каждый атрибут в отношении является атомарным.

– Вторая нормальная форма (2NF) утверждает, что непустые атрибуты должны функционально зависеть от всего ключа-кандидата.

– Третья нормальная форма (3NF) утверждает, что непрямые атрибуты должны напрямую (непереходно) зависеть от ключей-кандидатов.

Рассмотрение остальных NF не имеет большого смысла, т.к. в большинстве своём используются только первые три.

Есть 3 уровня представления информации, которые развились с течением времени: инфологический, дата логический и физический. На каждом из этих уровней приводиться структуризация информации, чтоб на третьем уровне информация была представлена в виде структур данных, которая уже должна быть реализована в памяти. Инфологический уровень определяет на какая информация о предметной области будет храниться в памяти компьютера, и в результате исследования предметной области составляется её модель, которая

с помощью терминов классов и объектов описывает предметную область. В инфологической модели представление идёт вне зависимости от того, что представляет собой сбор данных. Деталогическая и физическая модели реализуются непосредственно в БД, а именно в системах управления, физическая же модель определяет структуру хранения данных на физических носителях. Суть инфологического проектирования состоит в представление смысла предметной области. Чтобы описать предметную область наиболее часто используют модель “сущность - связь”, у ER – диаграммы модели есть лексикографическая структура, которая в свою очередь включает в себя текст и элементы графики. Семантическая (инфологическая) модель, на практике используется на первой стадии проектирования БД.

Процесс проектирования БД можно представить в виде последовательности переходов, от описания словесного (неформального) предметной области к формализации объектов предметной области в термине модели.

Можно выделить следующие этапы проектирования:

- Физическое проектирование БД;
- Проектирование инфологической модели предметной области – частично

формализованное описание объектов предметной области в терминах некоторой

семантической модели, например, в терминах ER–модели;

- Даталогическое или логическое проектирование БД;
- Системный анализ и словесное описание информационных объектов предметной области;

ER – модель опирается на понятие сущность, атрибут, связь и предметная область должна быть представлена совокупностью сущностей с атрибутами, между которыми установлена связь. В ER – модели с объектами связаны понятия: тип – набор предметов, явлений, которые выступают как единое целое; экземпляр – конкретный элемент набора, который означает тип;

множество – конкретный набор экземпляров типа. Сущность – это объект, который может быть идентифицирован некоторым способом, который отличает его от других объектов. Тип сущности означает набор однородных сущностей некоторого типа. Набор сущностей – множество сущностей одного типа. Сущность фактически является множеством атрибутов, которые описывают свойства всех членов данного набора сущности. Тип сущности означает множество однородных объектов, а “экземпляр сущности” – конкретный объект из множества объектов. К примеру, сущность студент, которая представляет всех студентов вуза, а один из них, Иванов Иван Иваныч, и он не является сущностью, а он является реализацией и причём конкретной. Атрибуты у студента это: ФИО, номер группы, номер зачетной книжки, дата зачисления и др.

Связь – это поименованное отношение, имеющее место между двумя сущностями. Такая связь является бинарной в том смысле, что она имеет место между двумя поименованными сущностями или же имеет вид отношения сущности к самой себе. Каждая связь имеет два конца, каждый из которых обладает: именем, степенью (мощностью), признаком обязательности. Эти свойства используются для характеристики связи по отношению к каждой из участвующей в ней стороне. Каждая сущность должна иметь уникальный идентификатор (ключ), то есть должна быть уникально определена: каждый экземпляр сущности должен иметь ямное и недвусмысленное определение, позволяющее отличать его от других экземпляров той же сущности. Ключом может быть атрибут, комбинация атрибутов, комбинация связей или атрибутов и связей. Например, ключом для сущности Студент является атрибут № зачетной книжки.

На сегодняшний день существуют несколько нотаций для представления ER–

диаграмм:

1. Нотация Чена: сущность изображается прямоугольником, атрибут – овалом, соединенным со своей сущностью (идентифицирующий атрибут

подчеркнут), а связь – ромбом, соединенным со связываемыми сущностями. Вид линии в месте соединения с сущностью определяет кардинальность связи («воронья лапка» – M, «крест» – 1). Имена сущности, атрибута и связи располагаются внутри их изображений (рис. 14).

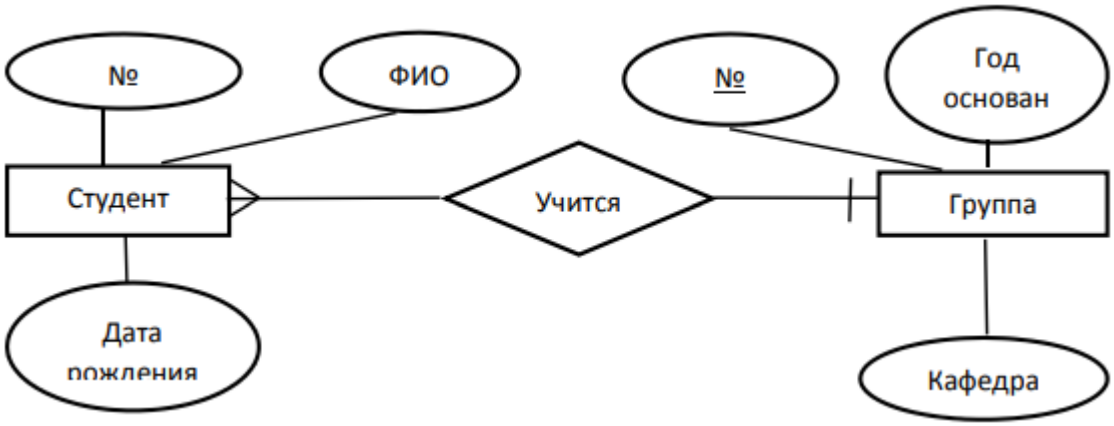

Рисунок 14 – Пример ER – диаграммы в нотации Чена.

Нотация Мартина: сущность изображается прямоугольником, внутри которого указано ее имя жирным шрифтом и список ее атрибутов (идентифицирующий атрибут подчеркнут), а связь – линией, название которой располагается над ней и ее вид в месте соединения с сущностью определяет кардинальность связи («воронья лапка» – M, «крест»– 1) (рис. 15).

Рисунок 15 - Пример ER – диаграммы в нотации Мартина.

Нотация Баркера: сущность изображается прямоугольником, внутри которого указано ее имя жирным и список ее атрибутов (перед идентифицирующим атрибутом стоит #), а связь – линией, название которой располагается над ней и ее вид в месте соединения с сущностью определяет кардинальность связи («воронья лапка» – M, отсутствие – 1) (рис. 16).

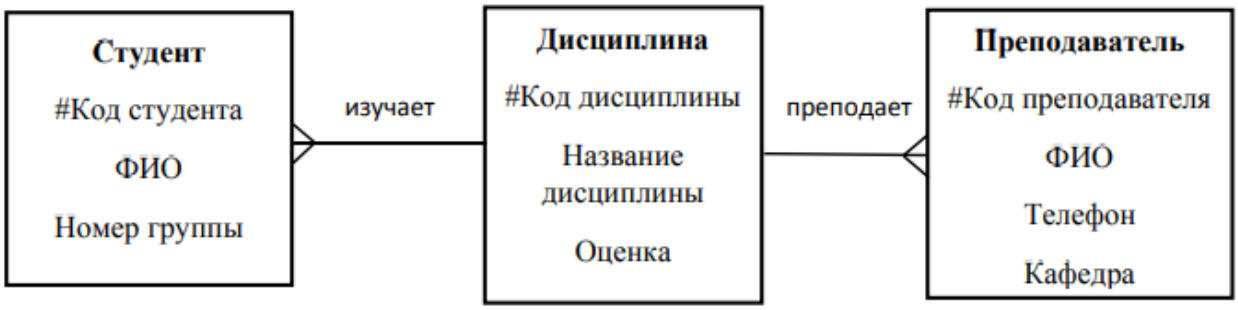

Рисунок 16 - Пример ER – диаграммы в нотации Баркера.

Нотация IDEFIX: сущность изображается прямоугольником, атрибут – овалом, соединенным со своей сущностью, а связь – ромбом, соединенным со связываемыми сущностями. Имена сущности, атрибута и связи располагаются внутри их изображений (рис. 17).

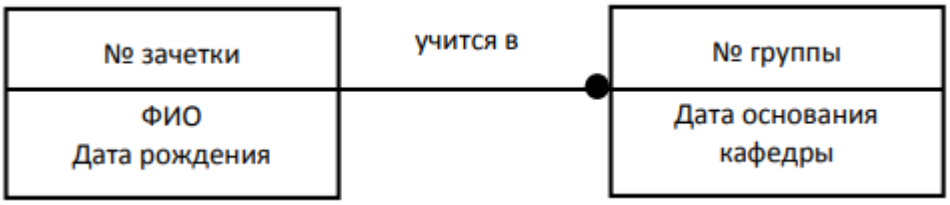

Рисунок 17 - Пример ER – диаграммы в нотации IDEFIX.

Нотация Бахмана: сущность изображается таблицей из одного столбца, столбцы которой являются атрибутами сущности (идентифицирующий атрибут выделен жирным шрифтом), а связь – стрелкой, соединяющей таблицы, направление которой указано на стороне M (рис. 18).

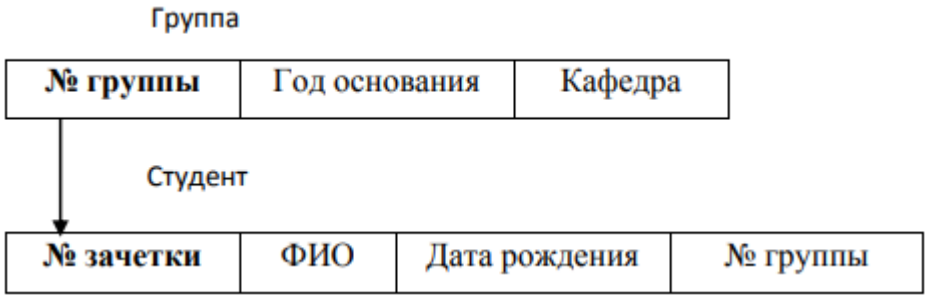

Рисунок 18 - Пример ER – диаграммы в нотации Бахмана.

Entity Relationship Diagram (ERD) - ER Diagram или ER model, представляет собой тип структурной диаграммы для использования при проектировании баз данных. ERD содержит различные символы и соединители, которые визуализируют две важные информации: основные сущности в области системы взаимосвязи между этими сущностями.

Путём объединения всех сущностей, а также после построения их связей, была создана ER-модель (рис. 19).

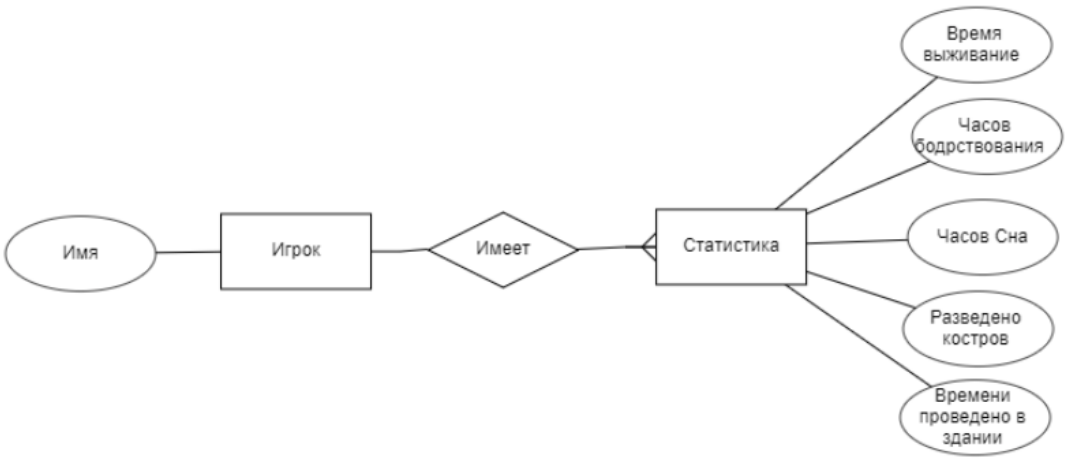

Рисунок 19 – ER-модель

1.4. Средства разработки базы данных

Определение СУБД.

Система управления базами данных (СУБД) — это программное обеспечение, предназначенное для хранения, извлечения, определения и управления данными в базе данных.

Некоторые типичные случаи использования ER:

Что такое СУБД?

Программное обеспечение СУБД в первую очередь функционирует как интерфейс между конечным пользователем и базой данных, одновременно управляя данными, механизмом базы данных и схемой базы данных, чтобы облегчить организацию данных и манипулирование ими.

Хотя функции СУБД сильно различаются, функции и возможности СУБД общего назначения должны включать: доступный для пользователя каталог, описывающий метаданные, систему управления библиотекой СУБД, абстракцию данных и независимость, безопасность данных, регистрацию и аудит активности, поддержку параллелизма и транзакций, поддержку для авторизации доступа, поддержки доступа из удаленных мест, поддержки восстановления данных СУБД в случае повреждения и принудительного

применения ограничений для обеспечения соответствия данных определенным правилам.

Выход СУБД — это встроенный в СУБД пакет SQL, который позволяет пользователю отображать отладочную информацию и выходные данные, а также отправлять сообщения из подпрограмм, пакетов, блоков PL/SQL и триггеров. Первоначально Oracle разработала пакет DBMS File Transfer, который предоставляет процедуры для копирования двоичного файла в базе данных или для передачи двоичного файла между базами данных.

Система управления базами данных функционирует за счет использования системных команд, сначала получая инструкции от администратора базы данных в СУБД, а затем соответствующим образом инструктируя систему либо извлекать данные, либо изменять данные, либо загружать существующие данные из системы. Популярные примеры СУБД включают облачные системы управления базами данных, системы управления базами данных в оперативной памяти (IMDBMS), системы управления столбцовыми базами данных (CDBMS) и NoSQL в СУБД.

Чем отличается СУБД от традиционной файловой системы?

Традиционная файловая система относится к ранним попыткам компьютеризировать ручную файловую систему. Файловые системы обычно используют устройства хранения, такие как CD-ROM или жесткий диск, для хранения и организации компьютерных файлов и данных с целью облегчения доступа.

Традиционная файловая система недорога, идеальна для небольшой системы с меньшим количеством частей, очень низкими усилиями по проектированию, изолированными данными и имеет простую систему резервного копирования, но небезопасна, имеет недостаток гибкости и множество ограничений, а также целостность. недостатки.

Преимущества СУБД по сравнению с традиционной файловой системой включают в себя: удобство для больших систем, возможность совместного использования данных, гибкость, целостность данных и сложную систему

резервного копирования. Требования к безопасности данных СУБД включают использование маскирования, токенизации, шифрования, списков контроля доступа, разрешений, брандмауэров и виртуальных частных сетей, что делает хранение данных и запросы в СУБД гораздо более безопасными, чем в традиционной файловой системе.

Дальше будут рассмотрены преимущества MsSql, СУБД которая была выбрана для этого проекта.

Установка упрощена

Он может быть установлен с помощью мастера установки, и необходимые обновления обнаруживаются и загружаются установщиком автоматически. Сложность установки программного обеспечения значительно минимизируется благодаря автоматической установке обновлений. Другие компоненты, такие как analytical и database Services, могут быть установлены отдельно. Автоматическое обновление также значительно снижает затраты на обслуживание.

Функции безопасности лучше

SQL Server 2008 использует управление на основе политик для обнаружения несоответствующих политик безопасности. Эта функция позволяет только авторизованному персоналу получать доступ к базе данных. Аудиты безопасности и события могут автоматически записываться в файлы журналов.

Повышенная производительность

MS SQL server имеет встроенную функцию прозрачного сжатия данных наряду с шифрованием. Пользователям не нужно изменять программы для шифрования данных. MS SQL server имеет контроль доступа в сочетании с эффективными инструментами управления разрешениями. Кроме того, он обеспечивает повышенную производительность, когда дело доходит до сбора данных.

Более низкая стоимость владения

SQL Server включает в себя эффективные инструменты управления

данными и интеллектуального анализа данных, а также разделение дисков. Оптимальное обслуживание вашего сервера можно обеспечить, следуя эффективным методам управления данными. Эти рекомендации также помогают обеспечить доступность и возможность восстановления данных.

Станьте сертифицированным гуру SQL с QuickStart. Запустите 7-дневную бесплатную пробную версию и получите полную команду по анализу, проектированию, разработке, запросам, внедрению или администрированию баз данных SQL.

1.5. Разработка базы данных

На основании разработанной модели системы было создано 2 таблиц в базе данных

При создании навой базы данных нужно придерживаться следующих шагов:

1) В обозревателе объектов выполняем подключение к экземпляру компонента Компонент SQL Server Database Engine и разворачиваем его (рис. 20);

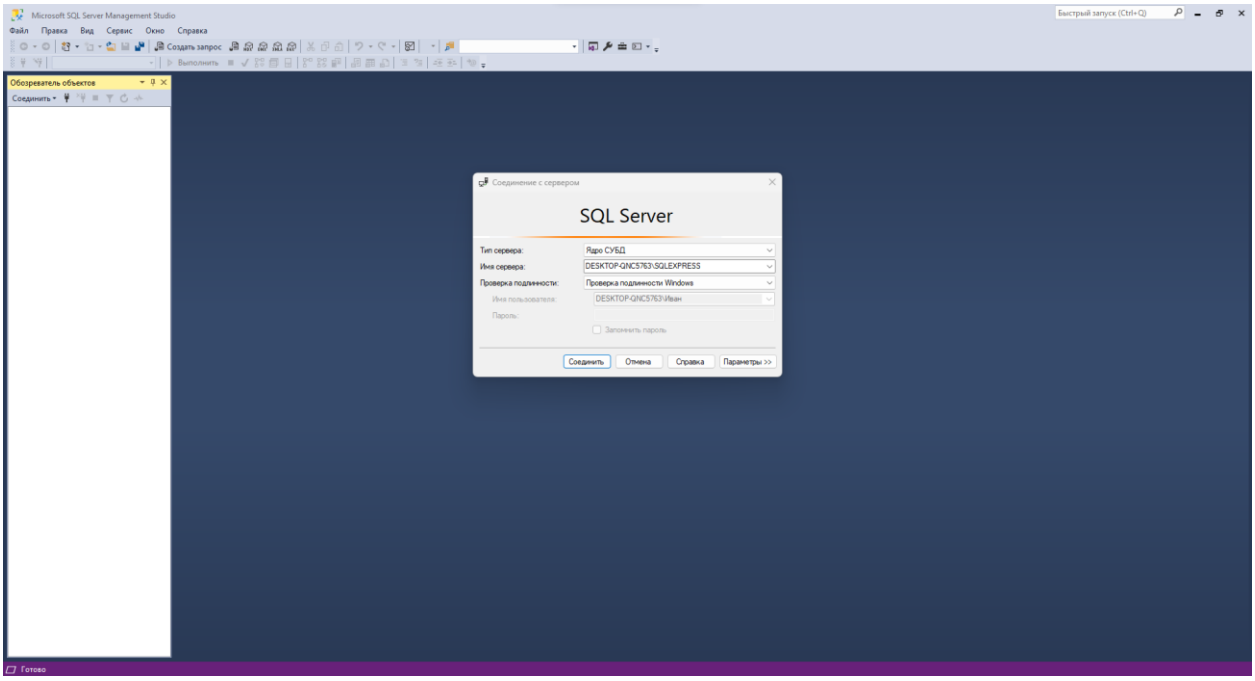

Рисунок 20 – Подключение к Microsoft SQL Server

2) После нужно выбрать правой кнопкой мыши элемент Базы данных,

после выбрать пункт Создать базу данных (рис. 21);

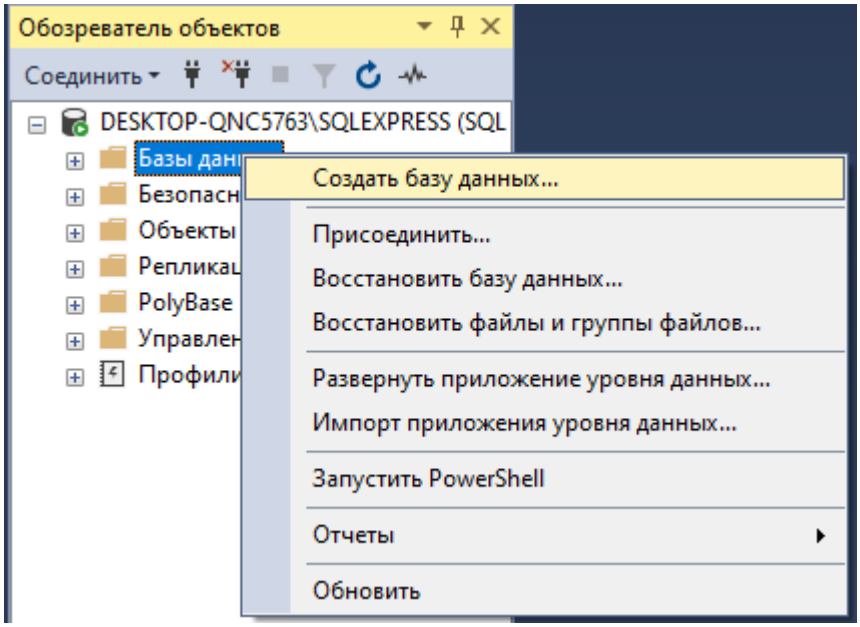

Рисунок 21 – Создание базы данных

3) Созданная база данных появляется в списке (рис. 22);

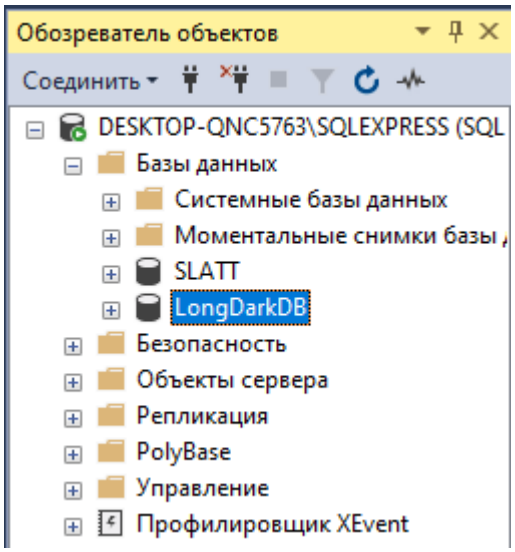

Рисунок 22 – Созданная база данных

Дальнейшим шагом является то что нужно добавить таблицы в базу данных. Для этого:

1) Нажимаем на папку “Таблицы” правой кнопкой мыши и в выпавшем меню выбираем “Создать – Таблица”;

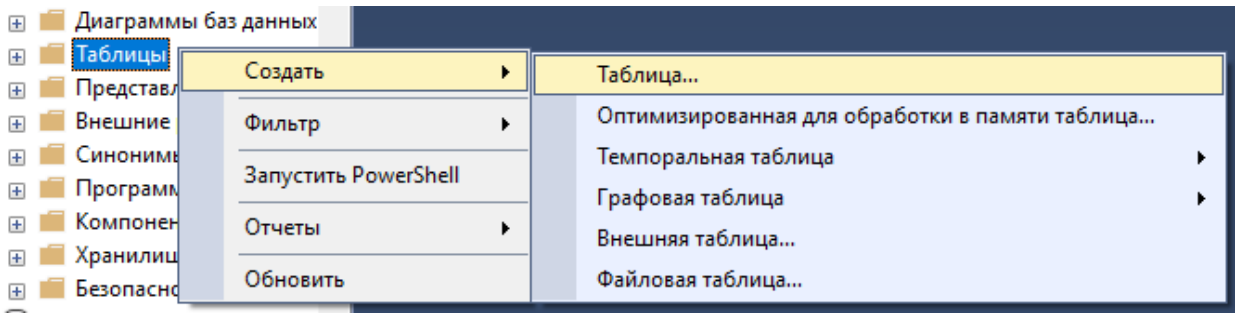

Рисунок 23 – Создание таблиц

2) Теперь нужно заполнить поля и типы данных в таблице для каждой сущности в соответствии с ER-моделью (рис. 24-25);

Имя столбца	Тип данных	Разрешить ...
Name	nvarchar(20)	☐
		☐

Рисунок 24 – Сущность “Игрок”

Имя столбца	Тип данных	Разрешить ...
SurvivalTime	time(7)	☐
WakingTime	time(7)	☐
SleepTime	time(7)	☐
BonfiresLit	int	☐
TimeSpentBuilding	time(7)	☐
		☐

Рисунок 25 – Сущность “Статистика Игрока”

3) После создания таблиц с первичными и внешними ключами, нужно будет создать и расставить связи, для этого нужно перейти в папку “Диаграммы” и “Создать диаграмму базы данных” (рис. 26)

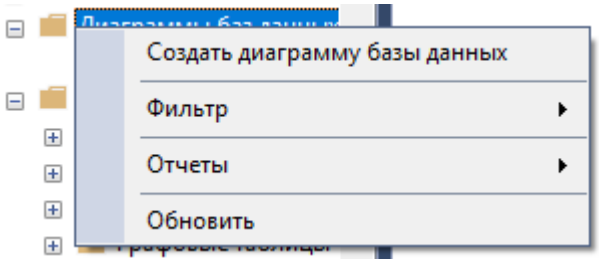

Рисунок 26 – Создание диаграммы баз данных

4) Теперь нужно добавить в диаграмму таблицы (рис. 27)

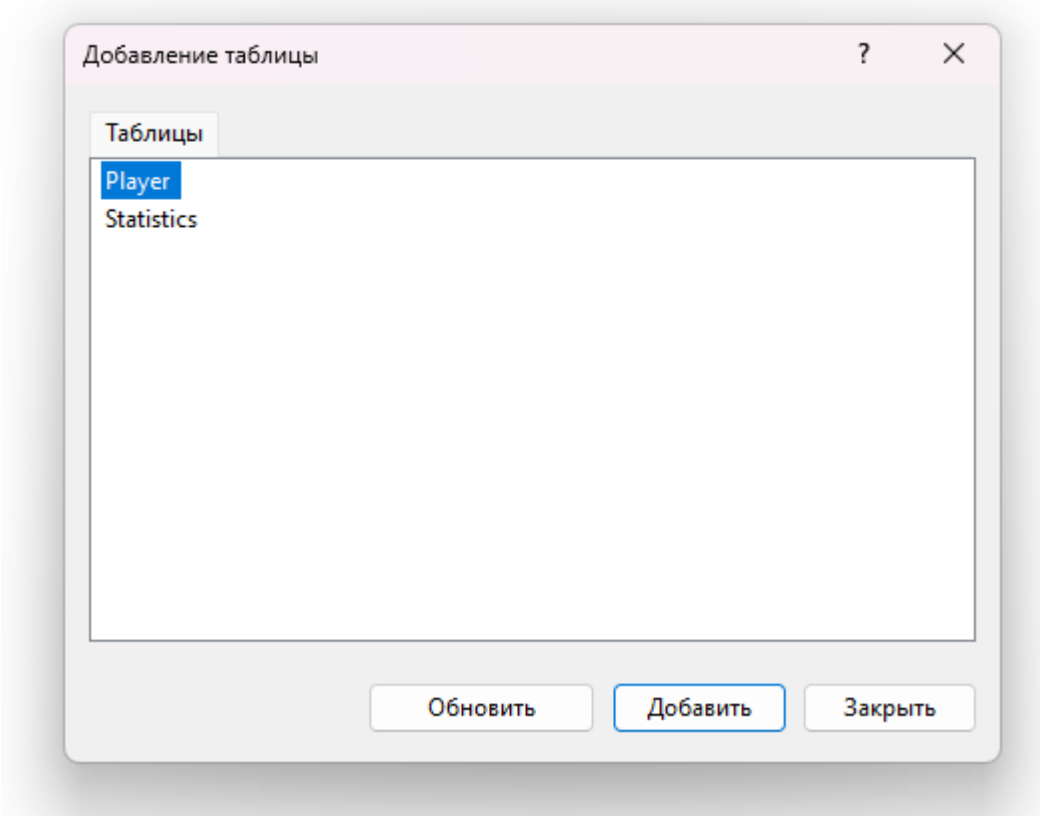

Рисунок 27 – Добавление таблиц в диаграмму

5) Соединяем первичный ключ одной таблицы с внешним ключом нужной таблицы (рис. 28)

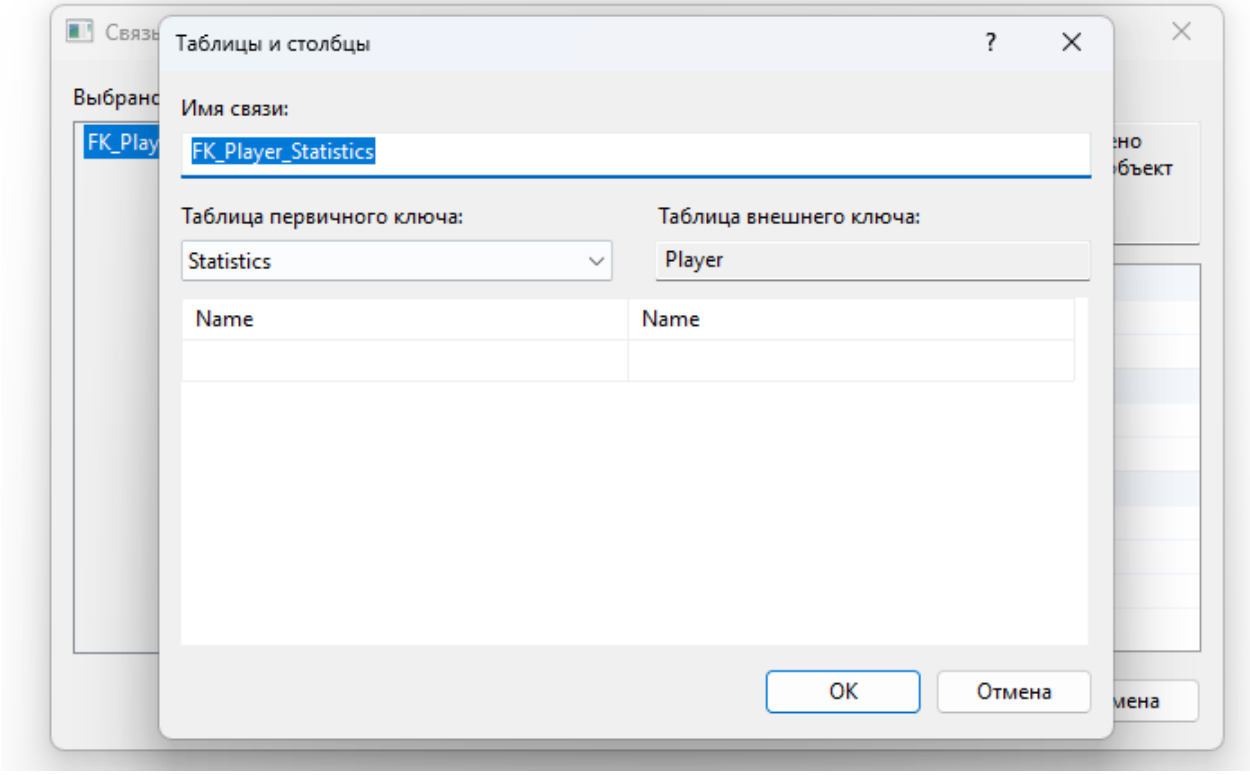

Рисунок 28 – Расставление связей

6) В результате завершения всех предыдущих этапов получаем готовую базу данных (рис. 29)

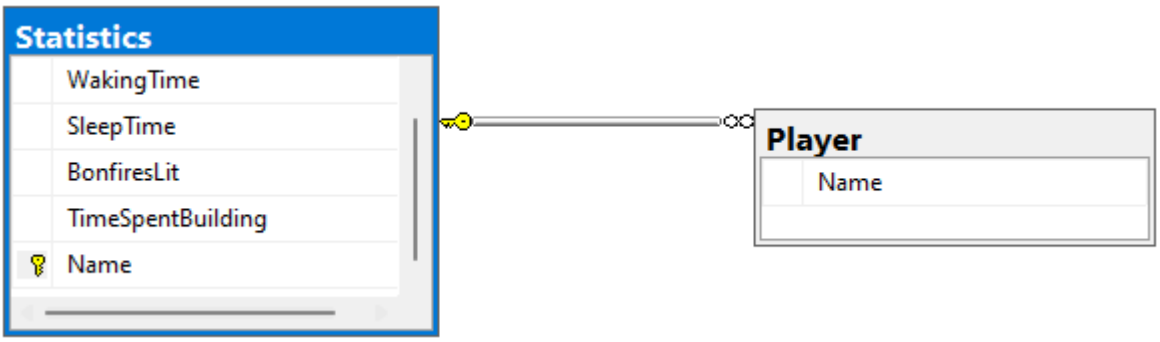

Рисунок 29 – Диаграмма базы данных

Запросы на выборку из базы данных:

- SELECT Name FROM Statistic – Вывести название всех лиц;
- SELECT Name FROM Player WHERE SleepTime > 4 – выбрать имена игроков, у которых время сна больше 4.

1.6 Руководство пользователя

После входа в игру открывается главное меню игры (рис. 30), в нём можно выбрать раздел настроек (рис. 31), где можно настроить громкость музыки и общую громкость. Вход в игру (рис. 32) осуществляется через кнопку Выживание или Сюжет в зависимости от режима, в который хочет поиграть пользователь. После входа в игру есть инвентарь, в котором есть текущие вещи игрока и его одежда.

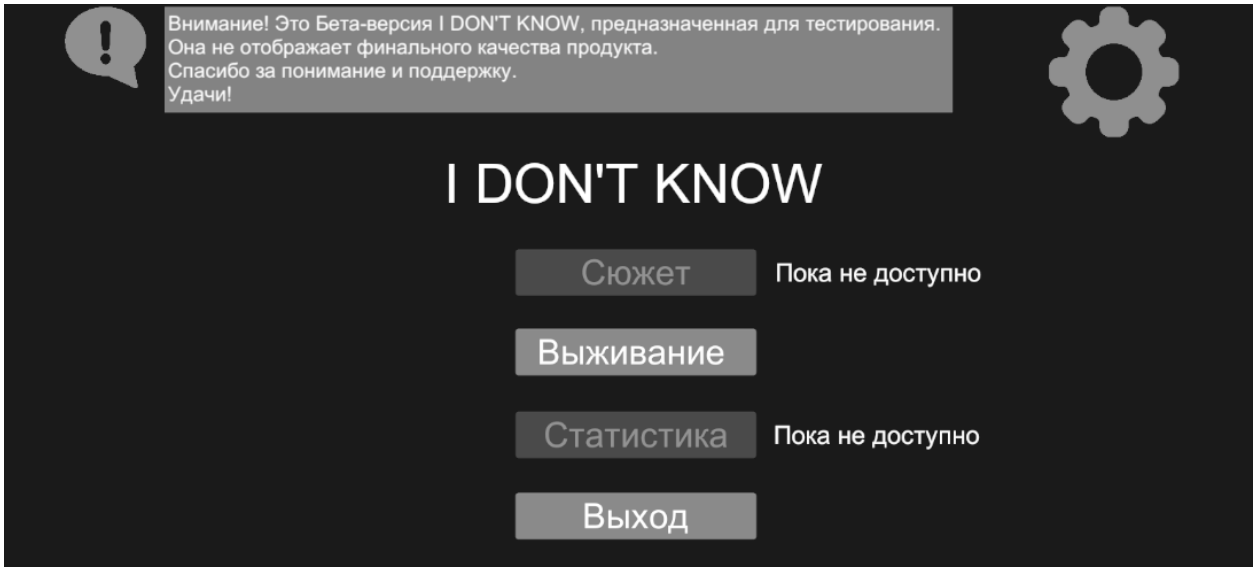

Рисунок 30 – Главное меню игры

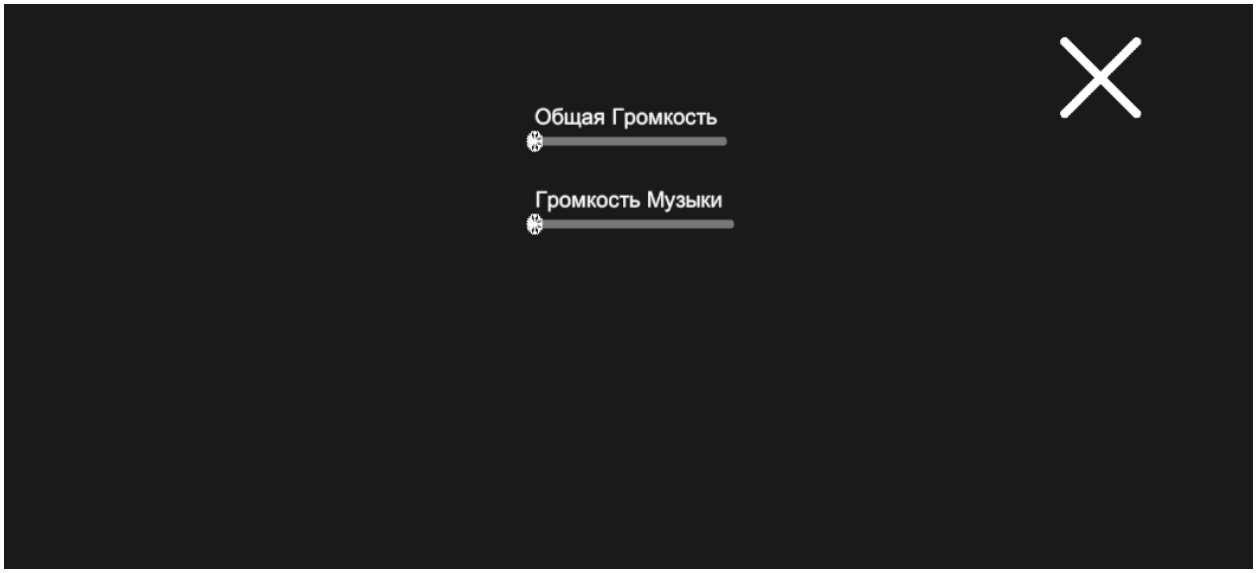

Рисунок 31 – Настройки

Рисунок 32 – Игра

СПИСОК ЛИТЕРАТУРЫ

1. Королев, Е. Н. Администрирование СУБД : учебное пособие для СПО / Е. Н. Королев, Б. Н. Тишуков, А. В. Мандрыкин. — Саратов : Профобразование, 2022. — 155 с.

2. Молдованова, О. В. Информационные системы и базы данных : учебное пособие для СПО / О. В. Молдованова. — Саратов : Профобразование, 2021. — 177 с.

3. Грошев, А. С. Основы работы с базами данных : учебное пособие для СПО / А. С. Грошев. — Саратов : Профобразование, 2021. — 255 с.

4. Суворова, Г. М. Основы информационной безопасности : учебное пособие для СПО / Г. М. Суворова. — Саратов : Профобразование, 2021. — 135 с.

5. Грекул, В. И. Управление внедрением информационных систем : учебное пособие для СПО / В. И. Грекул, Г. Н. Денищенко, Н. Л. Коровкина. — Саратов : Профобразование, 2021. — 277 с.

6. Швецов, В. И. Базы данных : учебное пособие для СПО / В. И. Швецов. — Саратов : Профобразование, 2019. — 219 с.

7. Абрамов, Г. В. Проектирование и разработка информационных систем : учебное пособие для СПО / Г. В. Абрамов, И. Е. Медведкова, Л. А. Коробова. — Саратов : Профобразование, 2020.

8. Стасышин, В. М. Разработка информационных систем и баз данных : учебное пособие для СПО / В. М. Стасышин. — Саратов : Профобразование, 2020. — 100 с.

9. Швецов, В. И. Базы данных : учебное пособие для СПО / В. И. Швецов. — Саратов : Профобразование, 2019. — 219 с.

10. Баженова, И. Ю. Основы проектирования приложений баз данных : учебное пособие для СПО / И. Ю. Баженова. — Саратов : Профобразование, 2019. — 325 с.

11. Разработка и защита баз данных в Microsoft SQL Server 2005 : учебное пособие для СПО / . — Саратов : Профобразование, 2019. — 148 с.

12. Лазицкас, Е. А. Базы данных и системы управления базами данных : учебное пособие / Е. А. Лазицкас, И. Н. Загумённикова, П. Г. Гилевский. — 2-е изд. — Минск : Республиканский институт профессионального образования (РИПО), 2018.

13. Платунова, С. М. Администрирование сети Windows Server 2012 : учебное пособие по дисциплине «Администрирование вычислительных сетей» / С. М. Платунова. — Санкт-Петербург : Университет ИТМО, 2015. — 102 с.

14. Левчук, Е. А. Технологии организации, хранения и обработки данных : учебное пособие / Е. А. Левчук. — Минск : Вышэйшая школа, 2007. — 240 с.

15. Платунова, С. М. Администрирование вычислительных сетей на базе MS Windows Server® 2008 : учебное пособие по дисциплине «Администрирование вычислительных сетей» / С. М. Платунова. — Санкт-Петербург : Университет ИТМО, 2012. — 41 с.