Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики
им. А.Н. Тихонова

Департамент компьютерной инженерии

Базы данных

Домашнее задание

Тема: База данных участников Осенней Школы “Бизнеса в стиле .RU”

Петухов Пётр БИВ 225

Москва 2024

Содержание

Содержание **1**
Анализ предметной области **2**
Анализ информационных задач и круга пользователей системы **5**
Определение требований к операционной обстановке **7**
Требования к оборудованию 7
Требования к программному обеспечению 7
Требования к сети 8
Человеческие ресурсы 8
Техническое обслуживание и поддержка 8
Выбор СУБД и других программных средств **9**
Логическое проектирование реляционной БД **10**
ER-диаграмма системы 10
Преобразование ER–диаграммы в схему базы данных 11
Составление реляционных отношений 12
Нормализация полученных отношений (3НФ) 15
Определение дополнительных ограничений целостности 19
Описание групп пользователей и прав доступа 20
Реализация проекта базы данных 21
Создание таблиц 21
Создание представлений (готовых запросов) 29
Назначение прав доступа 30
Создание триггеров 33
Создание индексов 38
Разработка стратегии резервного копирования 40

Анализ предметной области

База данных создается для сбора и хранения в одном месте информации об участниках крупной студенческой организации Высшей Школы Экономики “Бизнес в стиле .RU” на ежегодном интенсиве - Осенняя Школа с целью централизации всей информации крупной студенческой организации в одном месте и получения быстрого доступа к хранимым данным, отслеживании статистики и проведение различной аналитики по собранной информации, что будет полезно для интенсивов будущих годов.

Описание особенностей системы:

— Все участники Осеннего Интенсива регистрируются через бота, откуда данные со всей их информацией попадает в таблицу с информацией об участниках.

— У каждого участника имеется свой уникальный ID, его ФИО, номер телефона, ссылки на личные страницы в ВК и ТГ, университет, факультет, образовательная программа, курс, запись о направлениях (из 4-х возможных, но можно выбирать несколько), на которые он подаётся.

— И отдельно есть список всех менторов, у которых есть свои уникальные ID, ФИО, ссылки на личные страницы в ВК и ТГ, все направления (их тоже может быть несколько), в которых ментор обучает студентов.

— На Осенней школе есть всего 4 направления - трека для обучения: Web-разработка, Менеджмент, Дизайн, Маркетинг.

— Каждый студент может участвовать в обучении сразу на нескольких направлениях.

— Один ментор может обучать студентов на разных треках.

— Один ментор на одном направлении обучает сразу нескольких студентов.

— У каждого из 4-х треков есть собственные подразделы, причём каждый участник имеет право выбрать несколько подразделов, но не имеет право не выбирать ничего.

— Каждый студент на направлении уникален и вместе с ним в направлении хранятся данные о его выбранных профилях(подразделе), количествах посещённых занятий, информация о личном менторе, запись на собеседование по окончании обучения.

— Менторы имеют право редактировать данные и добавлять информацию у студента о количестве посещений занятия, количестве сделанных домашних работ студентом, успешно/неуспешно пройденном обучении, информацию о менторе, обучающего в данный момент ученика.

Выделим сущности предметной области и их атрибуты (**идентифицирующие атрибуты** выделены полужирным шрифтом, *многозначные* – курсивом, составные подчеркнуты):

1) Люди
Атрибуты: **уникальный ID**, ФИО, номер телефона, ссылка на личную страницу ВК, ссылка на личную страницу ТГ, год поступления, университет, факультет, образовательная программа в университете, курс, статус (студент или руководитель).

Связи: являются Студентами или Руководителями.

2) Студенты
Атрибуты: дата начала обучения студента, дата конца обучения студента, результаты.

Связи: являются Людьми, имеют Руководителя, состоят в Направлении, проходят Темы.

3) Руководители
Атрибуты: дата назначения на должность (ментора, главы или CEO), дата снятия с должности.

Связи: являются Людьми, имеют Студентов, состоят в Направлении.

4) Направления

Атрибуты: название, год появления, год ликвидации, краткое описание направления, действует ли сейчас(да/нет).

Связи: проходится участвующими, имеют руководство, включают в себя темы направлений.

5) Темы
Атрибуты: название темы, краткое содержание, *домашнее задание*, количество посещённых занятий, сколько дз сдано, прошёл ли обучение до конца(да/нет), прошёл ли итоговое собеседование, дополнительные требования (поле для комментариев от ментора).

Связи: входят в направления, проходятся участвующими.

Анализ информационных задач и круга пользователей системы

Основные задачи, которые будут выполнять пользователи базы участников интенсива Осенней Школы от “Бизнес в Стиле.Ру”:

1. Просмотр информации об участнике: любой участник может зайти и посмотреть информацию о себе - какие данные он ввёл в систему, какие у него на текущий момент направления обучения, сколько занятий по каждому направлению он посетил и сколько домашних работ по каждому направлению он сдал.

2. Добавление участников: Пользователи, имеющие на то права, могут добавлять и убирать участников осеннего интенсива из системы.

3. Автоматическая смена прав участников: Если студент успешно закончил интенсив и хочет стать ментором, у него автоматически повышаются права до пользователя “ментор”. Также и с менторами, которые успешно обучили учеников и хотят стать главой своего направления.

4. Обновление данных о прогрессе студентов: Любой ментор имеет право просмотреть всех своих учеников и обновить актуальную информацию по участию студента в занятиях, количестве выполненных домашних работ и выбранном для данного трека поднаправлении, смог ли студент успешно завершить курс и пройти финальное собеседование.

5. Обновление направлений на Осеннюю школу: Главный пользователь имеет право менять темы направлений и их поднаправлений каждый год, анализировать с помощью средств СУБД информацию обо всех участниках всех направлений по годам.

Системой будут пользоваться разные пользователи, у каждого из которых свой набор прав доступа:

1. Глава студ организации “Бизнес в Стиле. Ру”: это единственный пользователь, который имеет доступ к данным во всей базе данных,

возможность их просматривать и менять права пользователей. Главным образом назначает и снимает глав всех направлений, именно за счёт смены прав доступа у пользователя, создаёт новые и убирает старые направления в каждом году перед началом Осенней Школы, следит за текущей работой всей БД.

2. Глава направления: пользователь имеет право просматривать информацию обо всех участниках своего направления, не только о студентах, но и о менторах, а также изменять поля своего направления, назначать и снимать с должности менторов, но не может добавлять и убирать студентов.

3. Ментор: пользователь, выполняющий повседневные операции при работе со своими учениками, такие как просмотр, обновление и добавление информации о некоторых полях ученика(отметка о количестве посещённых занятий учеником, отметка о количестве выполненных пользователем домашних заданий, отметка от успешности/неуспешности пройденного курса, а также успешности/неуспешности пройденного собеседования. Есть доступ, чтобы просматривать себя и других менторов в системе, однако прав на то, чтобы просматривать НЕ своих учеников и изменять информацию о себе, у него нет.

4. Участник: имеет возможность один раз при регистрации на интенсив добавить информацию о своих персональных данных в БД, просматривать себя, свои направления и свою текущую информацию. Может менять только свои персональные данные, если они, например, были введены с ошибкой. Но не имеет доступа к информации о других участниках и тем более методам изменения чужих данных.

База данных со всей информацией об участниках интенсива Осенней Школы поможет повысить качество работы текущей системы хранения и обработки данных, обеспечивая безопасный и эффективный способ управления этими данными и связанными с ним задачами.

Определение требований к операционной обстановке

Требования к оборудованию

Требования к оборудованию для системы будут зависеть от размера базы и количества участников. Однако, как минимум, системе потребуется:

1. Сервер: Серверная машина с многоядерным процессором, достаточным объемом оперативной памяти (не менее 8 ГБ для небольших операций, масштабирование по мере необходимости) и достаточным пространством для хранения базы данных.

2. Клиентские персональные компьютеры: Это компьютеры, используемые студентами, менторами, главами направлений и главой своей студ организации для доступа к системе. Они должны обладать такой производительность, чтобы быть способными запускать современный веб-браузер и работать в нём.

Требования к программному обеспечению

1. Система управления базами данных (СУБД): Реляционная СУБД, такая как PostgreSQL, MySQL или Oracle, для управления базой данных. Выбор СУБД будет зависеть от таких факторов, как производительность, стоимость, масштабируемость и совместимость с другими системами.

2. Операционная система: Современная операционная система, такая как Linux, для сервисных и Windows или macOS для клиентских компьютеров. Выбор операционной системы может зависеть от используемой СУБД и другого программного обеспечения, а также от стоимости её развёртывания и разработки в её среде.

3. Веб-сервер: Программное обеспечение для размещения приложения, такое как Apache, Nginx или IIS от Microsoft.

4. Серверный язык/фреймворк: Это программное обеспечение, используемое для построения серверной логики системы. Например, фреймворки Node.js , Django (Python) или Ruby on Rails.

5. Фреймворк фронтенда: Программное обеспечение для создания пользовательского интерфейса, такое как React, Angular или Vue.js .

Требования к сети

1. Подключение к Интернету: надёжное и быстрое подключение к сети Интернет для удаленного доступа к системе. Требования к скорости будут зависеть от количества пользователей и объема передаваемых данных.
2. Безопасность: Меры сетевой безопасности для защиты данных во время передачи, такие как брандмауэры, SSL/TLS для безопасной передачи данных и VPN для удаленного доступа.

Человеческие ресурсы

1. Администратор базы данных: Квалифицированный персонал для управления СУБД, выполнения планового технического обслуживания и работы с резервным копированием баз данных.
2. Персонал ИТ-поддержки: лица, ответственные за управление сервером, сетью и клиентскими компьютерами, а также за оказание помощи пользователям в решении технических вопросов.
3. Разработчики: Команда для разработки, тестирования и сопровождения приложения. Они должны хорошо разбираться в выбранных серверных и интерфейсных языках/фреймворках, а также в SQL и используемых СУБД.

Техническое обслуживание и поддержка

Регулярное техническое обслуживание и обновления системы необходимы для обеспечения бесперебойной работы, безопасности и оптимизации производительности. Это включает в себя обновления СУБД, операционной системы, развёртывание новых версий системы и регулярные резервные копии баз данных.

В дополнение к этому, система должна соответствовать правилам защиты данных и конфиденциальности, которые могут потребовать специальных мер безопасности и повлиять на дизайн системы.

Выбор СУБД и других программных средств

В качестве СУБД было решено использовать систему PostgreSQL по следующим критериям:

- Соответствие ACID: полное соответствие PostgreSQL требованиям ACID (атомарность, согласованность, изоляция, долговечность) обеспечивает надёжную обработку транзакций, что является критическим требованием для крупных систем хранения и обработки данных.

- Масштабируемость: PostgreSQL предназначен для управления большими объемами данных и может эффективно масштабироваться по мере роста системы бронирования отелей (в том числе до многосерверных решений).

- Безопасность: PostgreSQL предлагает надёжные функции безопасности, включая надёжные механизмы контроля доступа, представления и детализированные разрешения, которые имеют решающее значение для многопользовательской среды, такой как наша система.

Для административного взаимодействия с базами данных в системе PostgreSQL применяется ПО pgAdmin 4.

В процессе исполнения проекта необходимо использовать систему Git для реализации функций совместной разработки и контроля версий.

Логическое проектирование реляционной БД

ER-диаграмма системы

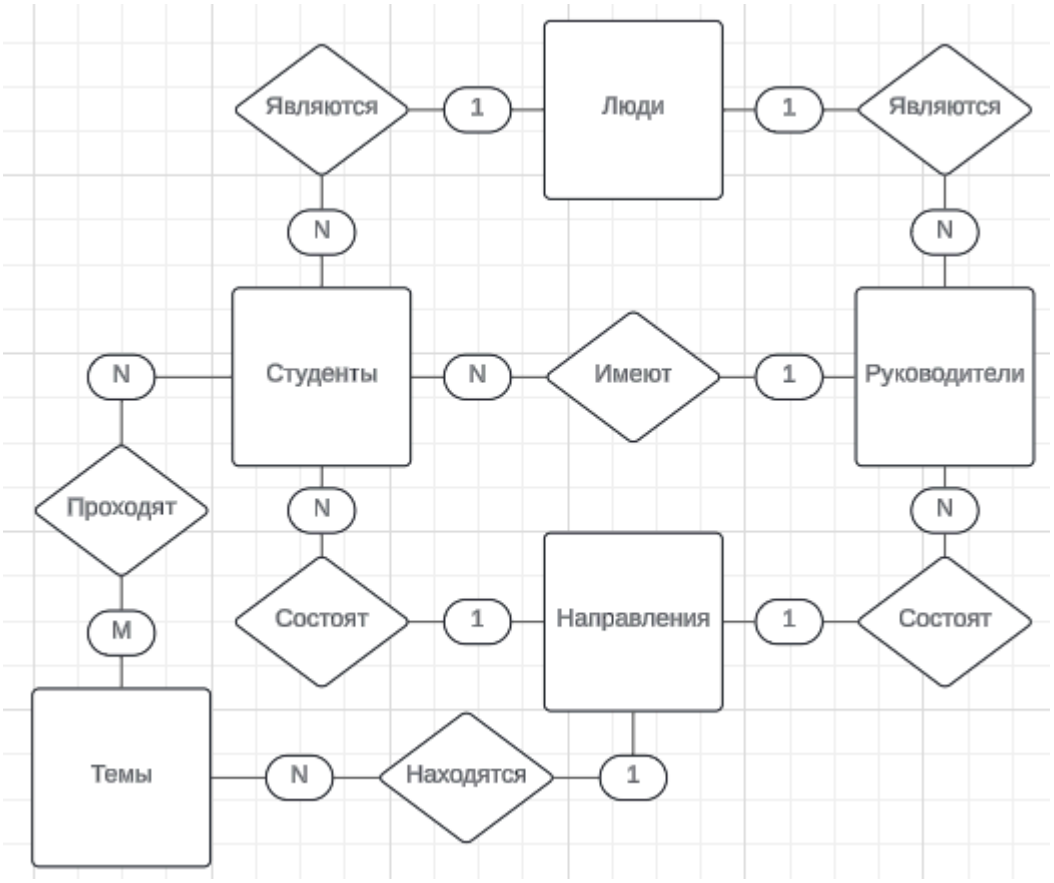

Рисунок 1 - ER-диаграмма системы

Преобразование ER–диаграммы в схему Базы данных

На основании ER-диаграммы построим схему базы данных. Все связи на данной диаграмме типа «один-ко-многим». В базе данных такие связи реализуются с помощью внешних ключей.

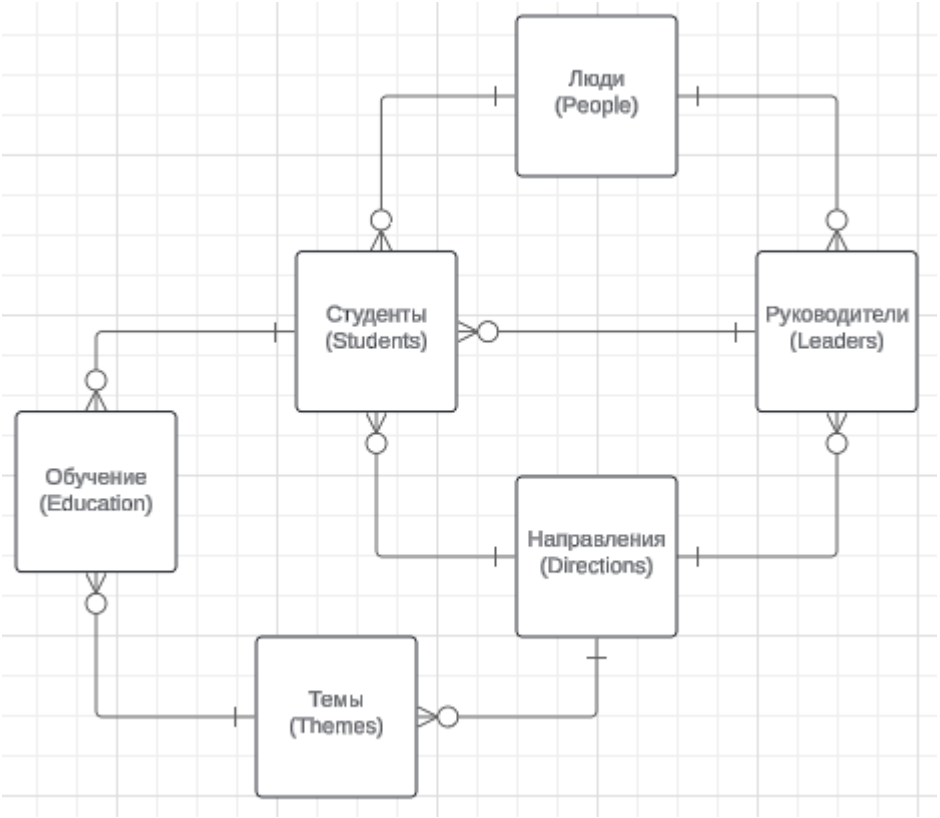

Рисунок 2 - Схема Базы данных

Составление реляционных отношений

Для каждого отношения указаны атрибуты с их внутренним названием, типом и длиной. Типы данных обозначаются так: N – числовой, C – символьный тип фиксированной длины, V – символьный тип переменной длины, D – дата, T – время.

Люди

Таблица 1 - Описание таблицы Люди

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер человека	id	N(6)	**уникальный первичный ключ**
ФИО	name	V(100)	обязательное составное поле
Телефон	phone	V(40)	обязательное поле, уникальное
Ссылка на личную страницу ВК	vk	V(100)	обязательное поле, уникальное
Ссылка на личную страницу ТГ	tg	V(100)	обязательное поле, уникальное
Год поступления	year	N(4)	обязательное поле
Университет	university	V(100)	
Факультет	faculty	V(100)	
Направление на факультете	direction	V(100)	
Курс	course	N(1)	
Статус	status	V(8)	обязательное поле, принимает значения “Студент” или “Руководитель”

Студенты

Для отношения *Студенты* введем суррогатный первичный ключ *Номер студента*, так как потенциальных первичных ключей у данной сущности нет.

Таблица 2 - Описание таблицы *Студенты*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер студента	id	N(6)	**суррогатный первичный ключ**
Номер направления	direction_id	N(6)	внешний ключ (к Направлениям)
Номер человека	student_id	N(6)	внешний ключ (к Людям), где status = “Студент”
Номер ментора	mentor_id	N(6)	внешний ключ (к Руководителям)

Дата начала обучения	start_date	D	обязательное поле
Дата окончания обучения	end_date	D	>=даты начала обучения
Результаты	results	V(50)	“Успешно прошёл обучение”, “Неудачно прошёл обучение” или “В процессе обучения”

Руководители

Для отношения *Руководители* введем суррогатный первичный ключ *Номер руководителя*, так как потенциальных первичных ключей у данной сущности нет.

Таблица 3 - Схема таблицы *Руководители*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер руководителя	id	N(6)	**суррогатный первичный ключ**
Номер направления	direction_id	N(6)	внешний ключ (к Направлениям)
Номер человека	leader_id	N(6)	внешний ключ (к Людям), где status = “Руководитель”
Должность	post	V(8)	обязательное поле, принимает значения “Ментор”, “Глава” или “CEO”
Дата назначения на должность	start_date	D	обязательное поле
Дата снятия с должности	end_date	D	>=даты назначения на должность

Направления

Для отношения *Направления* введем суррогатный первичный ключ *Номер направления*, так как потенциальный первичный ключ название может повторяться (например заново появилось направления, которое было ликвидировано).

Таблица 4 - Описание таблицы *Направления*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер направления	id	N(6)	**суррогатный первичный ключ**
Название	name	V(100)	обязательное поле, принимает значения: Web-разработка, Менеджмент, Дизайн или Маркетинг
Дата появления	start_date	D	обязательное поле
Дата ликвидации	end_date	D	>=даты появления

Краткое описание	description	V(200)	

Темы

Для отношения *Темы* введем суррогатный первичный ключ *Номер темы*.

Таблица 5 - Описание таблицы *Темы*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер темы	id	N(6)	**суррогатный первичный ключ**
Номер направления	direction_id	N(6)	внешний ключ (к Направлениям)
Название темы	name	V(100)	обязательное поле
Домашнее задание	homework	V(200)	обязательное поле
Краткое описание	description	V(200)	

Обучение

Для отношения Обучение введем суррогатный первичный ключ *Номер обучения*.

Таблица 6 - Описание таблицы Обучения

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер обучения	id	N(6)	**суррогатный первичный ключ**
Номер темы	theme_id	N(6)	внешний ключ (к Темам)
Номер студента	student_id	N(6)	внешний ключ (к Студентам)
Количество посещённых занятий	visits	N(4)	обязательное поле
Количество домашних заданий	number_of_homework	N(4)	обязательное поле
Пройдена ли тема	finish	C(3)	обязательное поле, “Да” или “Нет”
Дополнительные требования (от ментора)	additions	V(200)	

Нормализация полученных отношений до 3НФ включительно.

Люди

В отношении *Люди* присутствует составное поле *ФИО*. Разделим его на 3 поля: Фамилия, Имя, Отчество.

Вынесем информацию о университете, факультете, направлении и номере курса участника отдельно в отношения Университеты, Факультеты, Образовательные Программы.

Вынесем поле Статус, принимающее ограниченное число значений, отдельно в отношение Статусы.

Таблица 7 - Описание таблицы Люди

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер	id	N(6)	**уникальный первичный ключ**
Фамилия	surname	V(100)	обязательное поле
Имя	name	V(100)	обязательное поле
Отчество	patronymic	V(100)	
Телефон	phone	V(40)	обязательное поле, уникальное
Ссылка на личную страницу ВК	vk	V(100)	обязательное поле, уникальное
Ссылка на личную страницу ТГ	tg	V(100)	обязательное поле, уникальное
Год поступления	year	N(4)	обязательное поле
Номер образовательной программы	education_id	N(4)	внешний ключ (к Образовательным Программам на Факультетах)
Статус	status_id	C(1)	внешний ключ (к Статусы)

Таблица 8 - Схема таблицы *Образовательные Программы*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер образовательной программы	id	C(8)	**первичный ключ**
Название образовательной программы	direction	V(100)	обязательное поле

Таблица 9 - Схема таблицы *Образовательные Программы на Факультетах*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер образовательной программы на факультете	id	N(4)	**суррогатный первичный ключ**
Номер Образовательной программы	direction_id	C(8)	внешний ключ (к Образовательным программам)
Номер факультета	faculty_id	V(20)	внешний ключ (к Факультетам)

Таблица 10 - Схема таблицы *Факультеты*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Аббревиатура факультета	id	V(20)	**первичный ключ**
Название факультета	faculty	V(100)	обязательное поле
Номер университета	university_id	V(20)	внешний ключ (к Университетам)

Таблица 11 - Схема таблицы *Университеты*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Аббревиатура университета	id	V(20)	**первичный ключ**
Название университета	university	V(100)	обязательное поле

Таблица 12 - Схема таблицы *Статусы*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Название статуса	status	C(1)	**первичный ключ,** обязательное поле, где “C” - “Студент”, а “P” - “Руководитель”

Студенты

Вынесем поле Статусы прохождения, принимающее ограниченное число значений, отдельно в отношение Статусы прохождения.

Таблица 13 - Описание таблицы *Студенты*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер студента	id	N(6)	**суррогатный первичный ключ**

Номер направления	direction_id	N(6)	внешний ключ (к Направлениям)
Номер человека	student_id	N(6)	внешний ключ (к Людям), где status = “С”
Номер ментора	mentor_id	N(6)	внешний ключ (к Руководителям)
Дата начала обучения	start_date	D	обязательное поле
Дата окончания обучения	end_date	D	>=даты начала обучения
Номер статуса прохождения	progress_status_id	N(1)	внешний ключ (к Статусам прохождения)

Таблица 14 - Схема таблицы Статусы прохождения

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер статуса	id	N(1)	**суррогатный первичный ключ**
Статус	progress_statuses	V(30)	принимает значения “Успешно прошёл обучение”, “Неудачно прошёл обучение” или “В процессе обучения”

Руководители

Вынесем поле Должность, принимающее ограниченное число значений, отдельно в отношение Должности.

Таблица 15 - Схема таблицы *Руководитель*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер руководителя	id	N(6)	**суррогатный первичный ключ**
Номер направления	direction_id	N(6)	внешний ключ (к Направлениям)
Номер человека	leader_id	N(6)	внешний ключ (к Людям), где status = “Руководитель”
Номер должности	post_id	N(1)	внешний ключ (к Должностям)
Дата назначения на должность	start_date	D	обязательное поле
Дата снятия с должности	end_date	D	>=даты назначения на должность

Таблица 16 - Схема таблицы *Должности*

Содержание поля	*Имя поля*	*Тип, длина*	*Примечания*
Номер должности	id	N(6)	**суррогатный первичный ключ**
Должность	post	V(10)	обязательное поле, принимает значения “Ментор”, “Глава” или “CEO”

Схема базы данных после нормализации

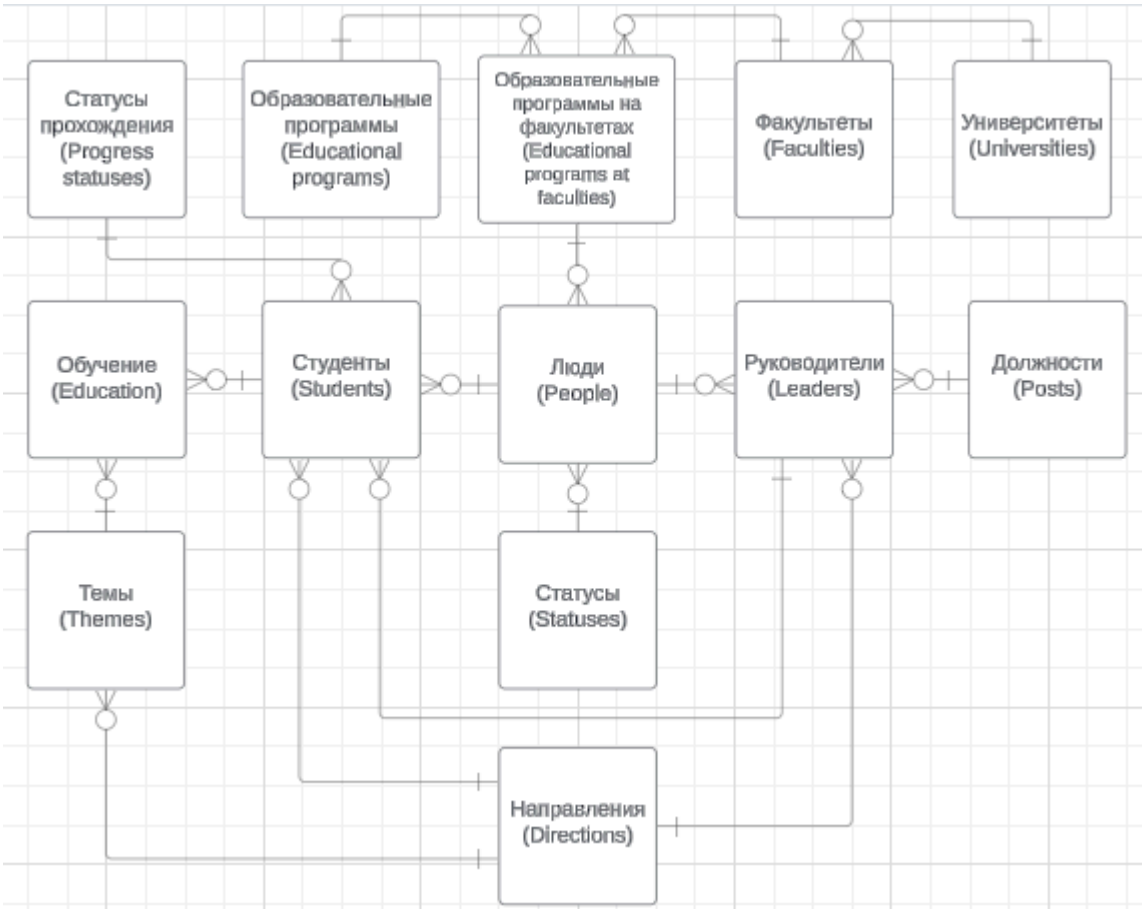

Рисунок 3 - Схема БД после нормализации

Определение дополнительных ограничений целостности

Пропишем дополнительные ограничения целостности, которые мы ранее не указали в примечаниях полей при составлении реляционных отношений:

1. Человек не может стать ментором, если он успешно не прошёл интенсив по своей специальности (успешность определяется значением строки “Успешно прошёл обучение” по полю Статус прохождения таблицы Обучение).
2. Человек не может стать своего главой направления, если он хотя бы год не был ментором этого направления (для этого ещё обязательно должен быть выполнен пункт 1).
3. Человек не может стать CEO студенческой организации “Бизнес в Стиле. Ру”, пока он хотя бы год не был главой одного из направлений на интенсиве (для этого ещё обязательно должны быть выполнены пункты 1 и 2).

Данные ограничения нельзя реализовать в схеме отношения при создании отношений и полей БД, поэтому их необходимо проверять через внешнее приложение, функцию, процедуру или специальную процедуру контроля данных – триггер.

Описание групп пользователей и прав доступа

Опишем для каждой группы пользователей права доступа к каждой таблице (Таблица 17). Права доступа должны быть распределены так, чтобы для каждого объекта БД был хотя бы один пользователь, который имеет право добавлять и удалять данные из объекта. Используются следующие сокращения:

Таблица 17 - Обозначения прав доступа

Обозначение	Расшифровка	Альтернативное обозначение
I (insert)	добавление данных	C (create)
S (select)	чтение данных	R (read)
U (update)	модификация данных	U (update)
D (delete)	удаление данных	D (delete)

Таблица 18 - Права доступа к таблицам для групп пользователей

Таблицы	Группы пользователей (роли)			
	Участник интенсива	Ментор	Глава направления	CEO
Люди	ISUD*	SUD*	SUD*	SUD*
Образовательные Программы	S	ISUD	S	S
Образовательные Программы на Факультетах	S	ISUD	S	S
Факультеты	S	ISUD	S	S
Университеты	S	ISUD	S	S
Статусы	S*	S	ISUD	ISUD
Студенты	S*	SU	ISUD	S
Статусы прохождения	S	S	ISUD	ISUD
Руководители		S	ISUD	ISUD
Должности		S	S	ISUD
Обучение	S	ISUD	ISUD	S
Темы	S	ISUD	ISUD	S
Направления	S	S	S	ISUD

* - только свои

Реализация проекта базы данных

Создание таблиц

1. Таблица “Направления”:

```
CREATE TABLE IF NOT EXISTS public."Directions"
(
    id numeric(6,0) NOT NULL,
    name character varying(100) COLLATE pg_catalog."default"
NOT NULL,
    start_date date NOT NULL,
    end_date date,
    description character varying(200) COLLATE
pg_catalog."default",
    CONSTRAINT "Directions_pkey" PRIMARY KEY (id),
    CONSTRAINT "CHK_name" CHECK (name::text =
'Web-разработка'::text OR name::text = 'Менеджмент'::text OR
name::text = 'Дизайн'::text OR name::text = 'Маркетинг'::text)
NOT VALID,
    CONSTRAINT "CHK_end_date" CHECK (end_date >= start_date)
NOT VALID
)
```

2. Таблица “Обучение”:

```
CREATE TABLE IF NOT EXISTS public."Education"
(
    id serial(6,0) NOT NULL,
    theme_id numeric(6,0),
    student_id numeric(6,0),
    visits numeric(4,0) NOT NULL,
    number_of_homework numeric(4,0) NOT NULL,
    finish character varying(3) COLLATE pg_catalog."default"
NOT NULL,
    additions character varying(200) COLLATE
pg_catalog."default",
    CONSTRAINT "Education_pkey" PRIMARY KEY (id),
    CONSTRAINT student_fkey FOREIGN KEY (student_id)
        REFERENCES public."Students" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
```

```
    CONSTRAINT theme_fkey FOREIGN KEY (theme_id)
        REFERENCES public."Themes" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "CHK_finish" CHECK (finish::text = 'Да'::text
OR finish::text = 'Нет'::text) NOT VALID
)
```

3. Таблица “Образовательные программы”:

```
CREATE TABLE IF NOT EXISTS public."Educational programs"
(
    id character varying(8) NOT NULL,
    direction character varying(100) COLLATE
pg_catalog."default" NOT NULL,
    CONSTRAINT "Educational programs_pkey" PRIMARY KEY (id)
)
```

4. Таблица “Образовательные программы на факультетах”:

```
CREATE TABLE IF NOT EXISTS public."Educational programs at
faculties"
(
    id numeric(4,0) NOT NULL,
    faculty_id character varying(20),
    direction_id character varying(8),
    CONSTRAINT "Educational programs at faculties_pkey"
PRIMARY KEY (id),
    CONSTRAINT direction_fkey FOREIGN KEY (direction_id)
        REFERENCES public."Educational programs" (id) MATCH
SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT faculty_fkey FOREIGN KEY (faculty_id)
        REFERENCES public."Faculties" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

5. Таблица “Факультеты”:

```
CREATE TABLE IF NOT EXISTS public."Faculties"
(
    id character varying(20) NOT NULL,
    faculty character varying(100) COLLATE
pg_catalog."default" NOT NULL,
```

```
    university_id character varying(20),
    CONSTRAINT "Faculties_pkey" PRIMARY KEY (id),
    CONSTRAINT university_fkey FOREIGN KEY (university_id)
        REFERENCES public."Universities" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

6. Таблица “Руководители”:

```
CREATE TABLE IF NOT EXISTS public."Leaders"
(
    id serial(6,0) NOT NULL,
    direction_id numeric(6,0),
    leader_id numeric(6,0),
    post_id numeric(1,0),
    start_date date NOT NULL,
    end_date date,
    CONSTRAINT "Leaders_pkey" PRIMARY KEY (id),
    CONSTRAINT direction_fkey FOREIGN KEY (direction_id)
        REFERENCES public."Directions" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT leader_fkey FOREIGN KEY (leader_id)
        REFERENCES public."People" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT post_id FOREIGN KEY (post_id)
        REFERENCES public."Posts" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT "CHK_end_date" CHECK (end_date >= start_date)
NOT VALID
)
```

7. Таблица “Люди”:

```
CREATE TABLE IF NOT EXISTS public."People"
(
    id numeric(6,0) NOT NULL,
    surname character varying(100) COLLATE
pg_catalog."default" NOT NULL,
    name character varying(100) COLLATE pg_catalog."default"
NOT NULL,
```

```
    patronymic character varying(100) COLLATE
pg_catalog."default",
    phone character varying(40) COLLATE pg_catalog."default"
NOT NULL,
    vk character varying(100) COLLATE pg_catalog."default" NOT
NULL,
    tg character varying(100) COLLATE pg_catalog."default" NOT
NULL,
    year numeric(4,0) NOT NULL,
    education_id numeric(4,0),
    status_id character varying(1),
    CONSTRAINT "People_pkey" PRIMARY KEY (id),
    CONSTRAINT education_fkey FOREIGN KEY (education_id)
        REFERENCES public."Educational programs at faculties"
(id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT status_fkey FOREIGN KEY (status_id)
        REFERENCES public."Statuses" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

8. Таблица “Должности”:

```
CREATE TABLE IF NOT EXISTS public."Posts"
(
    id numeric(6,0) NOT NULL,
    post character varying(10) COLLATE pg_catalog."default"
NOT NULL,
    CONSTRAINT "Posts_pkey" PRIMARY KEY (id),
    CONSTRAINT "CHK_post" CHECK (post::text = 'Ментор'::text
OR post::text = 'Глава'::text OR post::text = 'CEO'::text) NOT
VALID
)
```

9. Таблица “Статусы прохождения”:

```
CREATE TABLE IF NOT EXISTS public."Progress status"
(
    id numeric(1,0) NOT NULL,
    progress_statuses character varying(30) COLLATE
pg_catalog."default" NOT NULL,
    CONSTRAINT "Progress status_pkey" PRIMARY KEY (id),
    CONSTRAINT "CHK_progress_statuses" CHECK
(progress_statuses::text = 'Успешно прошёл обучение'::text OR
progress_statuses::text = 'Неудачно прошёл обучение'::text OR
```

```
progress_statuses::text = 'В процессе обучения'::text) NOT
VALID
)
```

10. Таблица “Статусы”:

```
CREATE TABLE IF NOT EXISTS public."Statuses"
(
    status character varying(1) COLLATE pg_catalog."default"
NOT NULL,
    CONSTRAINT "Statuses_pkey" PRIMARY KEY (status),
    CONSTRAINT "CHK_status" CHECK (status::text = 'C'::text OR
status::text = 'P'::text) NOT VALID
)
```

11. Таблица “Студенты”:

```
CREATE TABLE IF NOT EXISTS public."Students"
(
    id serial(6,0) NOT NULL,
    direction_id numeric(6,0),
    student_id numeric(6,0),
    mentor_id numeric(6,0),
    start_date date NOT NULL,
    end_date date,
    progress_status_id numeric(1,0),
    CONSTRAINT "Students_pkey" PRIMARY KEY (id),
    CONSTRAINT direction_fkey FOREIGN KEY (direction_id)
        REFERENCES public."Directions" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT mentor_fkey FOREIGN KEY (mentor_id)
        REFERENCES public."Leaders" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT progress_status_fkey FOREIGN KEY
(progress_status_id)
        REFERENCES public."Progress status" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
    CONSTRAINT student_fkey FOREIGN KEY (student_id)
        REFERENCES public."People" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID,
```

```
    CONSTRAINT "CHK_end_date" CHECK (end_date >= start_date)
NOT VALID
)
```

12. Таблица “Темы”:

```
CREATE TABLE IF NOT EXISTS public."Themes"
(
    id numeric(6,0) NOT NULL,
    direction_id numeric(6,0),
    name character varying(100) COLLATE pg_catalog."default"
NOT NULL,
    homework character varying(200) COLLATE
pg_catalog."default" NOT NULL,
    description character varying(200) COLLATE
pg_catalog."default",
    CONSTRAINT "Themes_pkey" PRIMARY KEY (id),
    CONSTRAINT direction_fkey FOREIGN KEY (direction_id)
        REFERENCES public."Directions" (id) MATCH SIMPLE
        ON UPDATE NO ACTION
        ON DELETE NO ACTION
        NOT VALID
)
```

13. Таблица “Университеты”:

```
CREATE TABLE IF NOT EXISTS public."Universities"
(
    id character varying(20) NOT NULL,
    university character varying(100) COLLATE
pg_catalog."default" NOT NULL,
    CONSTRAINT "Universities_pkey" PRIMARY KEY (id)
)
```

Создание представлений (готовых запросов)

1. Получить ФИО всех студентов, успешно завершивших обучение. (можно добавить выбор по направлению)

```
CREATE OR REPLACE VIEW students_successful AS
SELECT p.surname, p.name, p.patronymic
FROM "People" p
JOIN "Students" s ON p.id = s.student_id
JOIN "Progress status" ps ON ps.id = s.progress_status_id
JOIN "Directions" d ON d.id = s.direction_id
WHERE progress_statuses = 'Успешно прошёл обучение';
```

2. Получить ФИО и должность всех руководителей. (можно добавить выбор по направлению, факультету и университету)

```
CREATE OR REPLACE VIEW leaders_and_post AS
SELECT p.surname, p.name, p.patronymic, po.post
FROM "People" p
JOIN "Leaders" l ON p.id = l.leader_id
JOIN "Educational programs at faculties" epf ON p.education_id
= epf.id
JOIN "Educational programs" ep ON epf.direction_id = ep.id
JOIN "Faculties" f ON epf.faculty_id = f.id
JOIN "Universities" u ON f.university_id = u.id
JOIN "Posts" po ON l.post_id = po.id;
```

3. Получить все темы и домашние задания. (можно добавить выбор по направлению)

```
CREATE OR REPLACE VIEW themes_and_homeworks AS
SELECT t.name, t.homework
FROM "Themes" t
JOIN "Directions" d ON t.direction_id = d.id;
```

4. Получить всех студентов, количество посещений на различных темам которых превышает среднее значение по всем строкам таблицы Обучение.

```
CREATE OR REPLACE VIEW average_visits_materials AS
SELECT *
FROM public."Education"
WHERE visits > (SELECT AVG(visits) FROM public."Education");
```

5. Получить ФИО студентов и темы их домашних заданий, которые они не выполнили.

```
CREATE OR REPLACE VIEW students_with_unfinished_homework AS
SELECT p.surname, p.name, p.patronymic
FROM "People" p
JOIN "Students" s ON p.id = s.student_id
WHERE s.progress_status_id = (
     SELECT id FROM public."Progress status"
     WHERE progress_statuses = 'Неудачно прошёл обучение')
          AND s.mentor_id = 1;
```

6. Получить количество студентов, обучающихся по направлениям.

```
CREATE OR REPLACE VIEW students_per_direction AS
SELECT COUNT(*)
FROM "Students" s
JOIN "Directions" d ON s.direction_id = d.id
GROUP BY d.name;
```

Назначение прав доступа к представлениям

Таблица 9. Права доступа к представлениям

Представления	Группы пользователей (роли)			
	Участник интенсива	Ментор	Глава направления	CEO
students_successful			S	
leaders_and_post				S
themes_and_homeworks	S	S	S	
average_visits_materials		S		
students_with_unfinished_homework		S		
students_per_direction			S	S

Назначение прав доступа к таблицам

Чтобы предоставлять разные права доступа в нашей БД, сперва создадим роли пользователей БД:

```
CREATE ROLE intensive_participant; -- участник интенсива
CREATE ROLE mentor; -- ментор
CREATE ROLE head_of_direction; -- глава направления
CREATE ROLE CEO; -- Глава Бизнес в Стиле. Ру
```

Теперь предоставим права доступа, описанные ранее в таблице 18, пользователям БД с помощью командой GRANT:

```
GRANT INSERT, SELECT, UPDATE, DELETE ON "People" TO intensive_participant;
GRANT SELECT, UPDATE, DELETE ON "People" TO mentor;
GRANT SELECT, UPDATE, DELETE ON "People" TO head_of_direction;
GRANT SELECT, UPDATE, DELETE ON "People" TO CEO;


GRANT SELECT ON "Educational programs" TO intensive_participant;
GRANT INSERT, SELECT, UPDATE, DELETE ON "Educational programs" TO mentor;
GRANT SELECT ON "Educational programs" TO head_of_direction;
GRANT SELECT ON "Educational programs" TO CEO;


GRANT SELECT ON "Educational programs at faculties" TO intensive_participant;
GRANT INSERT, SELECT, UPDATE, DELETE ON "Educational programs at faculties" TO mentor;
GRANT SELECT ON "Educational programs at faculties" TO head_of_direction;
GRANT SELECT ON "Educational programs at faculties" TO CEO;


GRANT SELECT ON "Faculties" TO intensive_participant;
GRANT INSERT, SELECT, UPDATE, DELETE ON "Faculties" TO mentor;
GRANT SELECT ON "Faculties" TO head_of_direction;
GRANT SELECT ON "Faculties" TO CEO;


GRANT SELECT ON "Universities" TO intensive_participant;
GRANT INSERT, SELECT, UPDATE, DELETE ON "Universities" TO mentor;
GRANT SELECT ON "Universities" TO head_of_direction;
```

```
    GRANT SELECT ON "Universities" TO CEO;


    GRANT SELECT ON "Statuses" TO intensive_participant;
    GRANT SELECT ON "Statuses" TO mentor;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Statuses" TO
head_of_direction;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Statuses" TO
CEO;


    GRANT SELECT ON "Students" TO intensive_participant;
    GRANT SELECT, UPDATE ON "Students" TO mentor;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Students" TO
head_of_direction;
    GRANT SELECT ON "Students" TO CEO;


    GRANT SELECT ON "Progress status" TO
intensive_participant;
    GRANT SELECT ON "Progress status" TO mentor;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Progress status"
TO head_of_direction;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Progress status"
TO CEO;


    GRANT SELECT ON "Leaders" TO mentor;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Leaders" TO
head_of_direction;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Leaders" TO CEO;


    GRANT SELECT ON "Posts" TO mentor;
    GRANT SELECT ON "Posts" TO head_of_direction;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Posts" TO CEO;


    GRANT SELECT ON "Education" TO intensive_participant;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Education" TO
mentor;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Education" TO
head_of_direction;
    GRANT SELECT ON "Education" TO CEO;


    GRANT SELECT ON "Themes" TO intensive_participant;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Themes" TO
mentor;
```

```
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Themes" TO
head_of_direction;
    GRANT SELECT ON "Themes" TO CEO;


    GRANT SELECT ON "Directions" TO intensive_participant;
    GRANT SELECT ON "Directions" TO mentor;
    GRANT SELECT ON "Directions" TO head_of_direction;
    GRANT INSERT, SELECT, UPDATE, DELETE ON "Directions" TO
CEO;
```

Создание триггеров

Создадим следующие для лучшей автоматизации и защищённости нашей БД:

1. Автоматическое добавление в таблицу Студенты id при добавлении строки в таблицу Люди с полем status_id, ссылающимся на строку в таблице Статусы с полем status со значением C:

```
CREATE OR REPLACE FUNCTION add_new_student()
RETURNS TRIGGER AS
$$
BEGIN
            IF NEW.status_id = (SELECT status FROM
public."Statuses" WHERE status = 'C') THEN
              INSERT INTO public."Students" (student_id,
start_date)
          VALUES (NEW.id, CURRENT_DATE);
      END IF;

      RETURN NEW;
END;
$$
LANGUAGE plpgsql;


CREATE OR REPLACE TRIGGER trigger_add_new_student
AFTER INSERT ON public."People"
FOR EACH ROW
EXECUTE FUNCTION add_new_student();
```

2. Автоматическое добавление в таблицу Руководители id при добавлении строки в таблицу Люди с полем status_id, ссылающимся на строку в таблице Статусы с полем status со значением P:

```
CREATE OR REPLACE FUNCTION add_new_leader()
RETURNS TRIGGER AS
$$
BEGIN
            IF NEW.status_id = (SELECT status FROM
public."Statuses" WHERE status = 'P') THEN
```

```
                    INSERT INTO public."Students" (leader_id,
start_date)
              VALUES (NEW.id, CURRENT_DATE);
          END IF;

          RETURN NEW;
      END;
      $$
      LANGUAGE plpgsql;


      CREATE OR REPLACE TRIGGER trigger_add_new_leader
      AFTER UPDATE OF status_id ON public."People"
      FOR EACH ROW
      EXECUTE FUNCTION add_new_leader();
```

3. Автоматическое добавление строк в таблицу Обучение при изменении поля direction_id в таблице Студенты. Поля должны содержать студента и все темы, которые включены в направление выбранном в поле direction_id:

```
      CREATE OR REPLACE FUNCTION change_direction()
      RETURNS TRIGGER AS
      $$
           DECLARE
           iterator integer;
           theme_id_arr numeric(6)[];
      BEGIN
          IF NEW.direction_id != OLD.direction_id THEN
           theme_id_arr = ARRAY(SELECT id FROM public."Themes"
WHERE direction_id = NEW.direction_id);
                FOR iterator IN 1..array_length(theme_id_arr,
1) LOOP
                    INSERT INTO public."Education" (theme_id,
student_id, visits, number_of_homework, finish)
                 VALUES (theme_id_arr[iterator], NEW.id, 0, 0,
'Нет');
                END LOOP;
          END IF;
          RETURN NEW;
      END;
      $$
      LANGUAGE plpgsql;


      CREATE OR REPLACE TRIGGER trigger_change_direction
      AFTER UPDATE OF direction_id ON "Students"
```

```
FOR EACH ROW
EXECUTE FUNCTION change_direction();
```

4. Автоматическое удаление поля в таблице Студенты и добавление в таблицу Руководители при изменении поля status_id, ссылающимся на строку в таблице Статусы с полем status со значения C на P:

```
CREATE OR REPLACE FUNCTION update_student_status()
RETURNS TRIGGER AS $$
BEGIN
            IF   (NEW.status_id   =   (SELECT   status   FROM
public."Statuses" WHERE status = 'P')
          AND    OLD.status_id    =    (SELECT    status    FROM
public."Statuses" WHERE status = 'C') ) THEN
             DELETE FROM "Students" WHERE student_id = NEW.id;
             INSERT INTO "Leaders" (leader_id, start_date)
             VALUES (NEW.id, CURRENT_DATE);
         END IF;
         RETURN NEW;
END;
$$ LANGUAGE plpgsql;


CREATE OR REPLACE TRIGGER trigger_update_student_status
AFTER UPDATE OF status_id ON "People"
FOR EACH ROW
EXECUTE FUNCTION update_student_status();
```

5. Автоматическое изменение поля Статус_прохождения при выполнении студентом всех тем в направлении:

```
CREATE OR REPLACE FUNCTION
change_students_progress_status()

RETURNS TRIGGER AS $$

BEGIN

    IF NEW.finish !=OLD.finish  THEN

     IF ( SELECT COUNT(theme_id) FROM public."Education"
WHERE student_id = NEW.student_id )

     = ( SELECT COUNT(id) FROM public."Themes"
```

```
        WHERE direction_id = ( SELECT direction_id FROM
    public."Students" WHERE id = NEW.student_id ) ) THEN

            UPDATE public."Students" SET progress_status_id

        =    ( SELECT id FROM public."Progress status" WHERE
    progress_statuses = "Успешно прошёл обучение");

    END IF;

     END IF;

    RETURN NEW;

END;

$$ LANGUAGE plpgsql;



CREATE OR REPLACE TRIGGER
trigger_change_students_progress_status

AFTER INSERT OR UPDATE OF finish ON public."Education"

FOR EACH ROW

EXECUTE FUNCTION change_students_progress_status();
```

6. При переходе студента в руководство, т.е попытке изменить статус человека на P, будет автоматически проверяться, что обучение пройдено успешно:

```
    CREATE OR REPLACE FUNCTION
check_before_update_student_status()
    RETURNS TRIGGER AS
    $$
    BEGIN
         IF     (NEW.status_id     =     (SELECT     id     FROM
public."Statuses" WHERE status = 'P')
         AND     OLD.status_id     =     (SELECT     id     FROM
public."Statuses" WHERE status = 'C') ) THEN
         IF     (     SELECT     progress_status_id     FROM
public."Students" WHERE student_id = NEW.id ) !=
         (   SELECT   id   FROM   "Progress   status"   WHERE
progress_statuses = 'Успешно прошёл обучение' )
```

```
          THEN RAISE EXCEPTION 'ERROR. This student didn''t
finish courses succesfully yet. Now, he can''t become a
leader.';
          END IF;
          END IF;
          RETURN NEW;
     END;
     $$
     LANGUAGE plpgsql;


     CREATE            OR            REPLACE            TRIGGER
trigger_check_before_update_student_status
     BEFORE UPDATE OF status_id ON "People"
     FOR EACH ROW
     EXECUTE FUNCTION check_before_update_student_status();
```

Создание индексов

Система автоматически создаёт индексы по первичным ключам и уникальным полям. В дополнение к ним создадим индексы ко всем внешним ключам:

1. `CREATE INDEX idx_student_id ON Education(student_id);`
2. `CREATE INDEX idx_theme_id ON Education(theme_id);`
3. `CREATE INDEX idx_room_direction_id ON "Educational programs at faculties"(direction_id);`
4. `CREATE INDEX idx_faculty_id ON "Educational programs at faculties"(faculty_id );`
5. `CREATE INDEX idx_university_id ON Faculties(university_id);`
6. `CREATE INDEX idx_direction_id ON Leaders(direction_id);`
7. `CREATE INDEX idx_leader_id ON Leaders(leader_id);`
8. `CREATE INDEX idx_post_id ON Leaders(post_id);`
9. `CREATE INDEX idx_education_id ON People(education_id);`
10. `CREATE INDEX idx_status_id ON People(status_id);`
11. `CREATE INDEX idx_direction_id ON Students(direction_id);`
12. `CREATE INDEX idx_mentor_id ON Students(mentor_id);`
13. `CREATE INDEX idx_progress_status_id ON Students(progress_status_id);`
14. `CREATE INDEX idx_student_id ON Students(student_id);`
15. `CREATE INDEX idx_direction_id ON Themes(direction_id);`

Также добавим индексы для ускорения наших запросов:

1. Составной индекс для руководства для быстрого поиска по списку всех студентов:

```
CREATE INDEX idx_full name ON People(surname, name,
patronymic)
```

2. Составной индекс, который будет использоваться учащимися для быстрого поиска всей учебной информации для них в таблице “Обучение”:

```
CREATE INDEX idx_participants_information ON
Education(visits, number_of_homework, finish, additions)
```

3. Индекс для быстрого поиска должности человека в руководстве:

```
CREATE INDEX idx_post ON Leaders(post)
```

4. Составной индекс, который будет использоваться студентами для быстрого поиска тем и домашних заданий к темам в процессе обучения на направлении:

```
CREATE INDEX idx_homework_materials_for_theme ON
Theme(name, homework)
```

Разработка стратегии резервного копирования

Разработанная база данных является высоконагруженной системой в летний период перед подготовкой к интенсиву и в осенний период в момент проведения школы. Для обеспечения сохранности и целостности данных в этот период рекомендуется выполнять резервное копирование изменений в базе данных один- два раза в день в 12:00 дня и 12:00 ночи.

В остальные же периоды система будет использоваться редко, так что выполнять её резервное копирование надо будет раз в неделю.