Частное учреждение профессионального образования
«Высшая школа предпринимательства»
(ЧУПО «ВШП»)

КУРСОВАЯ РАБОТА

«Разработка базы данных для онлайн-платформы дистанционного обучения»

Выполнил:
студент 3-го курса специальности
09.02.07 «Информационные системы и программирование»
Зайцев Ярослав Сергеевич
подпись: _____

Проверил:
преподаватель дисциплины
«Численные методы»
колледжа ЧУПО «ВШП»
Ткачёв П.С
оценка: _____
подпись: _____

Тверь, 2025 г.

Оглавление

Введение

Актуальность

В наше время высоких технологий и стремительного развития онлайн сервисов появились и довелись до ума возможности дистанционного обучения. Которые крайне упрощают жизнь, теперь для получения определённых знаний не требуется куда-то уезжать и любую информацию можно получить прямо из дома, занимаясь напрямую с профессионалами. Данную тему можно считать актуальной и интересной, по скольку многие процессы дистанционного обучения актуальны и в других информационных сферах.

Определение цели работы

Для начало стоит внести ясность в тему и правильно сформировать цель курсовой работы. Темой является: «Разработка базы данных для онлайн-платформы дистанционного обучения». Исходя из этого цель работы - разработать базу данных для онлайн платформы дистанционного обучения. Прежде чем перейти к формулировки задачи, стоит разобраться в дистанционном обучении как явлении.

По версии сайта открытой онлайн энциклопедии ru.wikipedia.org, дистанционное обучение - образовательный процесс с применением совокупности телекоммуникационных технологий, имеющих целью предоставление возможности обучаемым освоить основной объём требуемой им информации без непосредственного контакта обучаемых и преподавателей в ходе процесса обучения.[2]

Если рассматривать, как онлайн платформы, то это способ получить информация используя телекоммуникационные сервисы, например Zoom, Google Meets. Онлайн платформы играют ключевую роль в организации дистанционного обучения, обеспечивая доступ к учебным материалам, интерактивным заданиям и обратной связи с преподавателями. Такие платформы, как Moodle, Coursera, Stepik и другие, позволяют организовать процесс обучения в удобной и доступной форме, поддерживая модульную систему, видеолекции, тесты и дискуссионные форумы.

Краткая история дистанционного обучения

Первый формы дистанционного обучения сформировались в XVIII века. Тогда британский учёный-стенограф Исаак Питман организовал дистанционное обучение по почте (так называемое корреспондентское обучение). Он полагал, что доступ к получению высшего образования должен быть обеспечен для всех желающих, независимо от их финансового достатка, национальности и вероисповедания. Следующий вклад в дистанционное обучения внесла Анна Тинкор, продолжавшая систему «корреспондентского обучения» по почте в конце XIX века. [1]

С развитием телекоммуникационных технологий таких как телефония, радио, телевиденье в начала и середине XX века, дистанционной обучения получило более широкое распространение, включая СССР, в виде теле, радиопередач, например «Чему и как учат в ПТУ». [1]

Важным этапом развития стал 1969 год. В Великобритании открыли The Open University – крупнейший центр дистанционного образования.

На территории России, дистанционное обучение начало развивается приблизительно в 1917 году, после конца революции. В рамках программ по ликвидации безграмотности выходили пособия для самообучения: «Школа на дому», «Рабочий техникум на дому», «Учись сам» и так далее. [1]

С появлением и развитием интернета, дистанционное обучение претерпело изменения. Интернет позволил упростить многие процессы, создавать онлайн курсы, лекции уроки, которые стали доступны по всему миру.

Стоит упомянуть важный этап популяризации дистанционного обучения. Пандемия COVID-19 стала глобальным катализатором для стремительного роста дистанционного образования. Закрытие школ и вузов по всему миру вынудило образовательные учреждения срочно внедрять онлайн-форматы, чтобы обеспечить непрерывность обучения. В этот период наблюдался резкий рост использования видеоконференцсвязи, электронных дневников, цифровых платформ и облачных сервисов.

Во многих странах, включая Россию, были разработаны национальные программы поддержки дистанционного образования. Преподаватели и студенты в

ускоренном режиме осваивали цифровые инструменты и платформы. Были запущены масштабные онлайн-курсы, разработаны рекомендации для эффективной организации учебного процесса в дистанционном формате. Появилось новое направление — смешанное обучение, совмещающее онлайн и очные элементы. В результате, несмотря на вызовы, пандемия дала мощный толчок цифровизации образования и сделала дистанционное обучение привычной и неотъемлемой частью образовательной системы.

Современное дистанционное обучение отличается следующими особенностями:

1. Доступность и гибкость. Обучающиеся могут осваивать материал в любое удобное время и из любой точки мира, имея лишь доступ в интернет.
2. Разнообразие платформ. Используются как международные, так и отечественные платформы (Coursera, Stepik, Сферум, РЭШ), что позволяет выбрать подходящий формат и язык преподавания.
3. Интерактивные форматы. Широко применяются видеолекции, тесты, вебинары, практические задания и симуляции, что делает процесс обучения более вовлекающим.
4. Поддержка обратной связи. Студенты могут взаимодействовать с преподавателями через чаты, форумы, электронную почту и видеосвязь, что позволяет быстро получать помощь и рекомендации.
5. Адаптивные технологии. Некоторые системы обучения подстраиваются под уровень знаний учащегося, предлагая индивидуальные траектории обучения.

Разница между традиционным и современным подходом очевидна: если раньше обучение было жёстко привязано ко времени и месту, то сегодня оно становится персонализированным, гибким и технологически продвинутым.

Постановка задач

Исходя из определения выше – сформулируем возможные задачи для достижения цели курсовой работы:

1. Анализ образовательных и бизнес-процессов в рамках дистанционного обучения.

2. Формулирование требований к создаваемой базе данных для поддержки дистанционного образовательного процесса.
3. Выбор подходящей системы управления базами данных (СУБД) для реализации проекта.
4. Построение логической и физической схемы базы данных.
5. Разработка набора взаимосвязанных таблиц базы данных на основе требований и особенностей дистанционного обучения.
6. Заполнение таблиц тестовыми образовательными данными (учебные курсы, студенты, преподаватели, задания и т.д.).
7. Реализация ключевых образовательных процессов с помощью выбранной СУБД (регистрация студентов, назначение заданий, ведение прогресса и т.д.).
8. Проведение тестирования работоспособности и корректности реализации процессов на тестовых данных.

Объект исследования

Объектом исследования является процесс проектирования и внедрения базы данных, предназначенной для обслуживания процессов дистанционного обучения.

Метод исследования

Методом исследования является моделирование, включающее анализ образовательной среды, структурирование пользовательского взаимодействия и формализацию требований к хранению и обработке данных в дистанционном обучении.

Первая глава

Определение бизнес-процессов условной онлайн-платформы дистанционного обучения

Для более полного понимания необходимо определить понятие "бизнес-процесса" в контексте дистанционного обучения.

Суть бизнес-процесса – это совокупность скоординированных активностей, имеющих конечной целью производство ценности для потребителя. В контексте электронного образования такими потребителями служат обучающиеся лица и преподаватели, а производимой ценностью – организация образовательных услуг высокого качества без географических ограничений.

В рамках образовательной деятельности также можно выделить три основных типа процессов:

1. Управляющие — процессы, связанные с организацией учебного процесса, управлением расписанием, формированием образовательной политики и стратегическим планированием.
2. Основные (операционные) — непосредственно реализующие обучение: регистрация студентов, организация доступа к курсам, выполнение заданий, сдача экзаменов, получение обратной связи и оценок.
3. Поддерживающие — процессы, обеспечивающие бесперебойную работу системы: техническая поддержка, учет данных, сопровождение пользователей, обновление учебных материалов и цифровой платформы.

Бизнес-процесс дистанционного обучения начинается с потребности обучающегося в освоении знаний и заканчивается достижением образовательных результатов, подтверждённых оценками, сертификатами или дипломами.

Каждый бизнес-процесс может быть разбит на более мелкие подпроцессы, которые легче анализировать и оптимизировать: например, процесс сдачи экзамена может включать регистрацию на экзамен, подготовку заданий, автоматическую проверку,

анализ результатов и уведомление студента. Такой подход позволяет улучшать эффективность и адаптивность системы дистанционного образования.

Процессы в системе дистанционного обучения должны быть организованы так, чтобы минимизировать излишние действия и обеспечивать максимальную ценность для обучающихся. Эффективно выстроенные процессы снижают затраты ресурсов, повышают удовлетворенность студентов и улучшают образовательные результаты.

В данном исследовании рассматривается условная образовательная платформа "EduPlus", которая должна быть масштабируемой, надёжной и универсальной. Все внутренние процессы платформы должны быть стандартизированы, обеспечивать быстрый отклик и точную обработку данных пользователей и преподавателей.

Ключевые бизнес-процессы платформы дистанционного обучения "EduPlus":

1. Хранение и обслуживание данных пользователей (обучающихся и преподавателей).
2. Регистрация студентов и преподавателей в системе.
3. Назначение курсов, модулей и заданий обучающимся.
4. Отслеживание прогресса, оценивание и хранение результатов обучения.
5. Предоставление обратной связи и отчётов для пользователей.

Эти процессы будут учтены при формировании требований к базе данных и проектировании её архитектуры.

Формулировка требований к разрабатываемой базе данных

Чтобы грамотно спроектировать базу данных для платформы дистанционного обучения, важно определить, какие требования предъявляются к такой системе хранения данных.

Согласно рекомендациям, приведённым в [3], все требования можно условно разделить на две большие категории:

1. Функциональные требования — описывают, что должна делать база данных.

   В контексте дистанционного обучения это может быть:

   - хранение информации о курсах, пользователях, оценках и заданиях;

- реализация механизма регистрации и авторизации;

- отслеживание прогресса студента;

-предоставление доступа к материалам и автоматическая проверка заданий.

2. Нефункциональные требования — описывают, как система должна выполнять свои функции. Это включает:

- надёжность хранения данных;

- масштабируемость при росте числа пользователей;

- обеспечение безопасности персональных данных;

- возможность одновременного доступа множества пользователей (асинхронные запросы);

- совместимость с внешними сервисами (например, платёжные или почтовые системы).

Бизнес-требования к базе данных помогают:

- согласовать цели заказчика и разработчика;

- установить чёткие технические задачи;

- обеспечить надлежащее качество реализации образовательного сервиса.

Исходя из вышеуказанного, можно выделить конкретные требования к базе данных системы "EduPlus":

1. Использование реляционной модели. Табличная структура с чёткими связями между сущностями (курсы, студенты, преподаватели) обеспечивает упорядоченное и масштабируемое хранение информации. SQL позволяет точно и гибко обрабатывать такие данные.
2. Наличие таблиц для ключевых компонентов: пользователей, учебных материалов, обратной связи, системы контроля прогресса.
3. Корректная организация связей между таблицами, что позволяет реализовать логику образовательных процессов, например, чтобы студент не мог получить доступ к модулю без прохождения предыдущего.
4. Поддержка высокой нагрузки и множества одновременных подключений. Для этого база данных должна быть легко масштабируемой и иметь механизмы кэширования и оптимизации запросов.

В качестве альтернативы реляционным базам можно рассматривать NoSQL-решения (MongoDB, Cassandra и др.), которые подходят для хранения больших объёмов неструктурированных данных. Однако, учитывая строгую структуру образовательного процесса и важность связей между элементами (например, студент — курс — оценка), наиболее рациональным выбором остаётся реляционная СУБД (например, PostgreSQL или MySQL).

Учитывая сформулированные требования, следующим шагом будет построение логической модели базы данных, соответствующей описанным процессам платформы "EduPlus".

Выбор СУБД для реализации базы данных

Для реализации данной базы данных онлайн платформы дистанционного обучения «EduPlus» была выбрана СУБД MySQL. Этот выбор обоснован рядом преимуществ, которые соответствуют требованиям, как и техническим, так и бизнес-требованиям проекта:

1. Производительность и масштабируемость данных. MySQL эффективно обрабатывает большие объемы информации и поддерживает множество одновременных подключений, что важно для систем с высоким количеством пользователей, но не без ограничений, в случае данного проекта это не критично.

2. Поддержка реляционной модели. MySQL идеально подходит для структурированных образовательных данных например: таблицы пользователей, курсов, заданий и оценок. Такие данные легко организуются и взаимосвязываются

3. Безопасность. Система предоставляет огромный инструментарий для осуществления безопасности данных. Например, для разграничения доступа, аутентификации шифрования и аудита - что является крайне важным для хранения персональных данных обучающихся.

4. Гибкость и функциональность. MySQL поддерживает хранимые процедуры, триггеры, различные типы данных, индексов и обеспечивает возможность глубокой работы с данными. Также легкость импорта в языки программирования, например Python, Java и так далее.

5. Доступность, простота и открытый исходный код. MySQL – бесплатное решение с огромным сообществом, что облегчает работу с СУБД, поддержку и адаптацию под нужды платформы.

6. Кроссплатформенность. MySQL работает на большинстве современных стационарных ОС, что делает её удобным и универсальным решением.

Области применения MySQL в проекте «EduPlus:

- Учёт студентов и преподавателей;
- Хранения структуры курсов и учебных материалов;
- Фиксация результатов обучения и оценивания;
- Формирования аналитики

Хотя у MySQL есть некоторые ограничения (например, по работе со сложными временными или географическими данными или могут возникнуть проблемы с безопаснотью), в рамках задач образовательной платформы эти ограничения не критичны. В случае необходимости в будущем возможен переход на более специализированные решения или использование гибридного подхода.

Стоит поговорить и о возможных минусах данной СУБД:

1. Безопасность. Как и любая СУБД, MySQL не может быть защищена на 100% и в случае реальной базы данных могут потребоваться дополнительные меры безопасности.

2. Ограниченность типов данных. Некоторые другие СУБД имеют намного больший набор типов данных, что добавляет им универсальности, например временные данные, но в случае данной работы это не является существенным недостатком

3. Возможные проблемы с производительностью. При обработке огромных потоков данных, расширяемой масштабируемостью могут возникнуть серьёзные проблемы производительности вплоть до замедления работы или вовсе отказа, утери данных и уничтожения базы данных. В данном случае, данный минус не актуален, так как работа хоть и отражает почти полный функционал базы данных

для платформы онлайн обучения, но всё же она не имеет такого объема информации.

Таким образом, MySQL полностью соответствует текущим потребностям платформы "ЭдуПлюс" и предоставляет надёжную основу для построения и масштабирования образовательной базы данных.

Почему MySQL, а не альтернативы?

Выбор системы управления базами данных (СУБД) — критически важный этап в разработке платформы дистанционного обучения. При оценке различных вариантов для реализации проекта "EduPlus" рассматривались как реляционные, так и не реляционные решения. На основе анализа было принято решение в пользу MySQL. Ниже представлено подробное обоснование этого выбора, а также сравнение с альтернативами.

Основными конкурентами в выборе СУБД были PostgeSQL и нереляционные СУБД, такие как MongoDB, Cassandra, Redis или Neo4j.

Почему не PostgreSQL?

PostgreSQL - мощная и надёжная СУБД, однако в контексте проекта "EduPlus" она имеет ряд особенностей, которые делают её менее приоритетной:

1. PostgreSQL может требовать больше ресурсов при больших объемах одновременных запросов

2. PostgreSQL имеет более сложные настройки для создания, развертывания базы данных

3. Избыточный функционал, например расширенные типы данных, пространственные индексы, они не будут использованы в функционале платформы

Почему не NoSQL?

Нереляционные СУБД, такие как MongoDB, Cassandra, Redis или Neo4j, также были рассмотрены. Однако у них есть ряд недостатков в контексте образовательной системы:

1. MongoDB - не поддерживает сложные связи между документами на уровне СУБД. В результате логика работы системы (например, последовательное прохождение модулей курса) должна реализовываться вручную, что увеличивает сложность разработки.

2. Redis - отличное решение для кэширования и хранения временных данных, но не предназначено для долговременного хранения структурированных данных.

3. Cassandra - подходит для обработки больших потоков данных, но не имеет нативной поддержки SQL и ограничена в функциональности для транзакций.

4. Neo4j - графовая СУБД, ориентированная на другие задачи (например, социальные графы), и требует специфической модели хранения, неприменимой к типичным структурам курса.

Примеры использования в онлайн платформе дистанционного обучения

1. Регистрация и учет пользователей: Система должна обеспечивать быстрое добавление новых студентов, преподавателей и администраторов, а также хранить историю их активности, прохождения курсов, выполнения заданий и получения сертификатов.

2. Управление образовательным контентом: Хранение информации о курсах, темах, модулях, материалах, тестах и заданиях. MySQL позволяет эффективно организовать структуру образовательных данных и легко управлять их изменениями.

3. Финансовые операции: Учёт оплаты за курсы, подписки, скидки, промокоды и возвраты. MySQL обеспечивает надёжную обработку транзакций и хранение финансовых данных с соблюдением принципов согласованности и целостности.

4. Обслуживание и техническая поддержка: Хранение истории обращений в службу поддержки, регистрации ошибок, анализа обратной связи от пользователей. Это позволяет анализировать качество платформы и оперативно реагировать на проблемы.

Выбор MySQL в качестве основной СУБД для платформы «EduPlus» обусловлен рядом причин. Например, простата установки и администрирования, высокая производительность при большом количестве запросов, легкая интеграция и распространённость. Благодаря использованию MySQL можно достичь эффективного управления структурированными данными образовательной платформы с минимальными затратами и высокой надёжностью

Краткий вывод

После анализа требований к онлайн-платформе «EduPlus» и возможных архитектурных решений, реляционная модель и функциональные возможности MySQL были признаны оптимальными. MySQL позволяет организовать стабильную, масштабируемую и безопасную работу системы, отвечающую всем ключевым бизнес-процессам: от регистрации пользователей до анализа учебной активности. На основе этих данных сформирована модель базы данных, максимально соответствующая задачам платформы онлайн-обучения.

Дополнительно MySQL обеспечивает:

1. централизованное хранение информации о курсах, пользователях и прогрессе;
2. быстрое выполнение запросов при построении отчётности и аналитики обучения;
3. гибкость в управлении правами доступа и разграничении ролей (преподаватель, администратор, слушатель);
4. устойчивую работу с возможностью масштабирования при росте числа пользователей и объёмов данных.

Таким образом, использование MySQL позволяет построить надёжную техническую основу для платформы дистанционного обучения, обеспечивая как повседневную работу пользователей, так и развитие функционала в будущем.

Вторая глава

Построение схемы базы данных.

Для работы с MySQL был выбран официальный инструмент MySQL Workbench

MySQL Workbench - инструмент для визуального проектирования баз данных, интегрирующий проектирование, моделирование, создание и эксплуатацию БД в единое бесшовное окружение для системы баз данных MySQL. Является преемником DBDesigner 4 от FabForce. [4]

Возможности MySQL WorkBench

MySQL Workbench предоставляет широкий спектр возможностей для проектирования баз данных в купе с достаточно простым, удобным интерфейсом и функционалом, позволяет выполнять любые задачи в сфере проектирования баз данных.

Если рассматривать сферу проектирования баз данных, то можно выделить обширный функционал например:

1. Средства для проектирования ER-диаграмм, с возможностями прямого проектирования и Reverse Engineer (обратного).
2. Простота изменений проектировочных данных.

Если рассматривать MySQL Workbench со стороны разработки, то можно привести примеры. Упрощения множества SQL запросов. Сама MySQL Workbench упрощает некоторые задачи с помощью визуальных инструментов, без обязательной работы с кодом. Также различные дополнительные средства по типу автозаполнении, выделения синтаксиса, подробный отчет об ошибке, хранение большого количество подключений локально

Стоит затронуть тему администрирования. MySQL Workbench предоставляет широкий спектр возможностей для администрирования базы данных. Администраторы и разработчики имеют доступ к визуальной консоли, которая может быть использована для просмотра всё базы данных. Еще одной

возможностью станет широкая возможность к настройке серверов и подключения, помимо этого есть еще огромное количество средств, например возможность резервного копирования данных и восстановления их.

Всё это при поддержке визуальных инструментов MySQL Workbench, что значительно облегчает задачи в любых сферах, начиная от проектирования, заканчивая сложной серверной работой.

ER-Диаграмма и набор таблиц проекта

По версии сайта ru.wikipedia.com ER-модель (от англ. Entity-Relationship model, модель «сущность — связь») — модель данных, позволяющая описывать концептуальные схемы предметной области. [4]

ER-модель (модель "Сущность-Связь") — это инструмент для начального планирования базы данных. Она помогает определить главные сущности в системе (например, Клиент, Заказ) и понять, как они связаны друг с другом (например, Клиент делает Заказ).

Позже, когда приходит время строить реальную базу данных, этот план (ER-схему) переводят в конкретные правила и таблицы под выбранный тип базы (реляционную, объектную и т.д.).

Сама ER-модель — это просто идея, набор правил. Её можно нарисовать, но рисовать не обязательно. Самый популярный способ нарисовать такую модель — диаграмма "сущность-связь" (ERD).

Часто люди путают термины:

ER-модель — это сама идея, план структуры данных.

ER-диаграмма (ERD) — это один из способов нарисовать этот план.

Фактически ER-диаграмма это разновидность блок-схем. Простыми словами данная схема отражает сущности внутри системы и как-либо связаны между собой для. Чаще всего применяется в проектировании баз данных в разных сферах от образовательных и вплоть до научных целей.

Перед построением определим таблицы который будут входить в перечисленные в прошлых главах возможности:

1.users

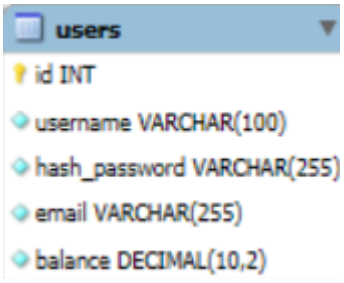

Рисунок 1. – Таблица студентов.

Сущность «пользователь» хранит данные для входа и работы с курсами.

1) id INT PRIMARY KEY AUTO_INCREMENT — уникальный идентификатор.

2) username VARCHAR(100) - логин, до 100 символов достаточно для удобства.

3) hash_password VARCHAR(255) - хэш пароля (bcrypt/scrypt), до 255 символов.

4) email VARCHAR(255) - адрес электронной почты для восстановления доступа.

5) balance DECIMAL(10,2) - текущий баланс пользователя (точное хранение денежных сумм).

2.teachers

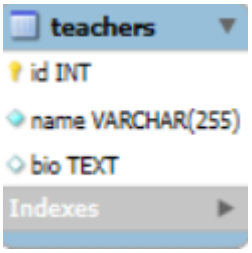

Рисунок 2. – таблица преподавателей.

Сущность «преподаватель» хранит информацию о лицах, которые ведут курсы.

1) id INT PRIMARY KEY AUTO_INCREMENT - уникальный идентификатор преподавателя.

2) name VARCHAR(255) - полное имя.

3) bio TEXT - текст биографии, опыта, достижений.

3.courses

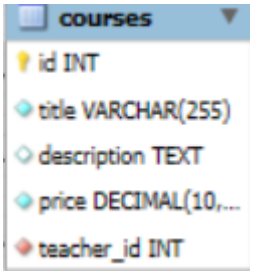

Рисунок 3. – Таблица курсов.

Сущность «курс» объединяет уроки под единым названием и ценой.

1) id INT PRIMARY KEY AUTO_INCREMENT - уникальный номер курса.

2) title VARCHAR(255) - заголовок курса.

3) description TEXT - развернутое описание целей и содержания.

4) price DECIMAL(10,2) - стоимость доступа к курсу.

5) teacher_id INT - внешняя ссылка на teachers.id, связывает курс с его автором.

4.lessons

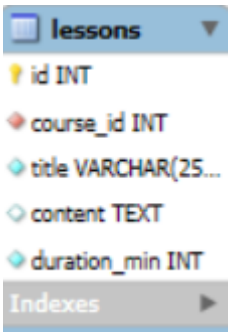

Рисунок 4. – Таблица уроков

Сущность «урок» описывает отдельные занятия внутри курса.

1) id INT PRIMARY KEY AUTO_INCREMENT - уникальный номер урока.

2) course_id INT - внешний ключ на courses.id, указывает, к какому курсу относится.

3) title VARCHAR(255) - заголовок урока.

4) content TEXT - основной материал (текст, ссылки, вёрстка).

5)duration_min INT - рекомендуемая длительность в минутах.

5.enrollments

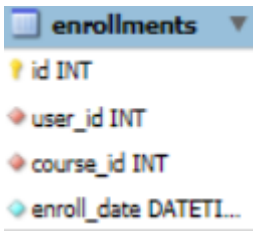

Рисунок 5. -Таблица записей на курс.

Сущность «запись на курс» фиксирует факт подключения пользователя к курсу.

1) id INT PRIMARY KEY AUTO_INCREMENT - идентификатор записи.

2) user_id INT - внешний ключ на users.id, кто записался.

3) course_id INT - внешний ключ на courses.id, на какой курс.

4) enroll_date DATETIME - дата и время подписки.

6.payments

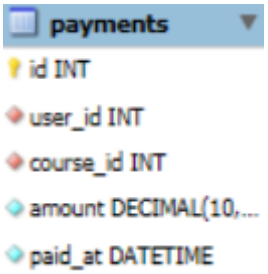

Рисунок 6. – Таблица платежей.

Сущность «платёж» хранит информацию о финансовых транзакциях.

1) id INT PRIMARY KEY AUTO_INCREMENT - номер транзакции.

2)user_id INT - внешний ключ на users.id, кто оплатил.

3) course_id INT - внешний ключ на courses.id, за что оплатил.

4) amount DECIMAL(10,2) - сумма платежа.

5) paid_at DATETIME - момент совершения оплаты.

7.sessions

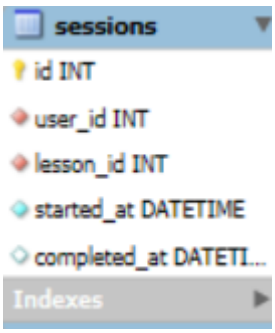

Рисунок 7. – Таблица сессий (фактических взаимодействий с уроками и курсами)

Сущность «сессия» фиксирует фактическое взаимодействие пользователя с уроком.

1) id INT PRIMARY KEY AUTO_INCREMENT - идентификатор сессии.

2) user_id INT - внешний ключ на users.id, кто работал.

3) lesson_id INT - внешний ключ на lessons.id, какой урок.

4) started_at DATETIME - время начала.

5) completed_at DATETIME - время окончания урока (может быть NULL, если не завершён).

Каждая связь один ко многим задаётся внешними ключами (user_id, teacher_id, course_id, lesson_id), что позволяет:

1) одному пользователю иметь много записей на курсы (enrollments), платежей (payments) и сессий (sessions);

2) одному курсу принадлежать много уроков и записей на курс;

3)одному уроку соответствовать множество сессий;

4)одному преподавателю вести несколько курсов.

Исходя из ранее применяемых формулировок приступим к построению ER диаграммы. Для лучшего вида сам рисунок перенесен на другую страницу.

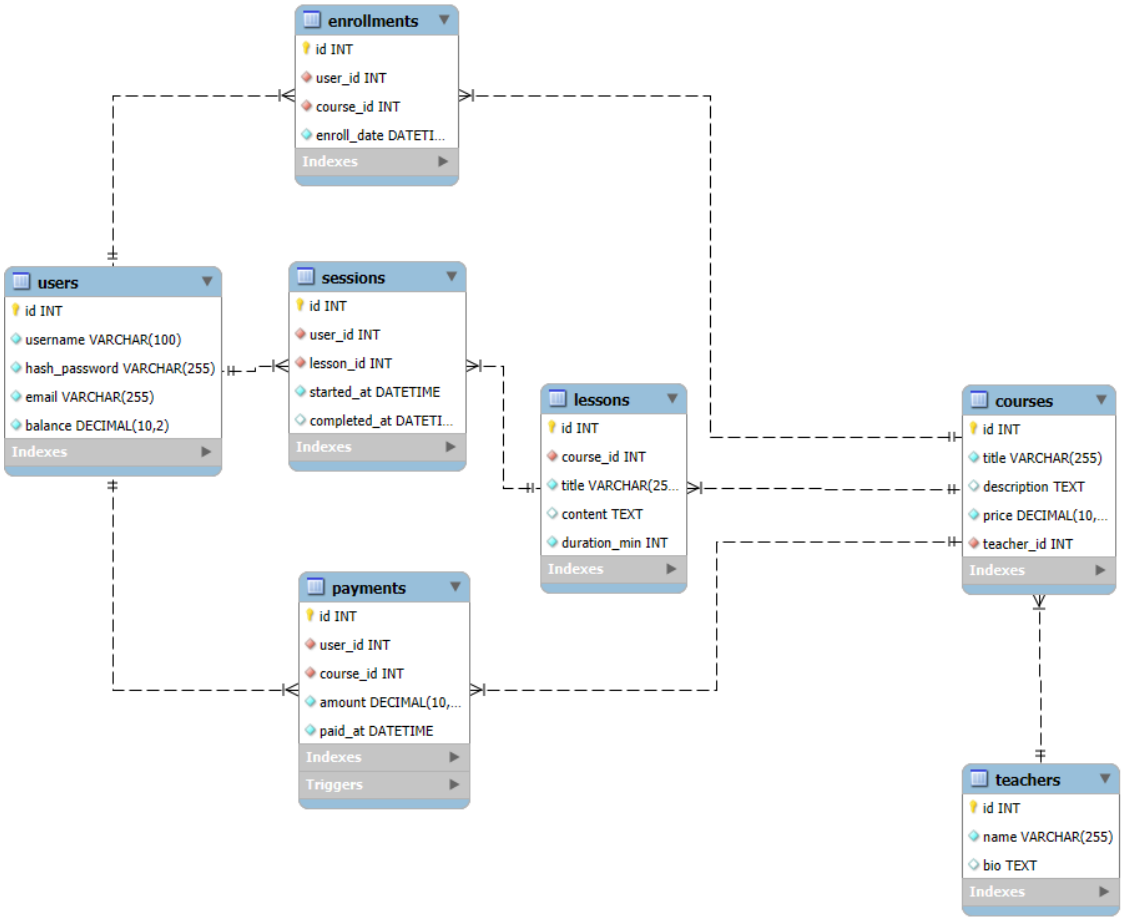

Рисунок 8. – ER-диаграмма базы данных для онлайн платформы дистанционного обучения.

Связи в проекте

Вся модель данных спроектирована с опорой на связь «один-ко-многим» — именно она лежит в основе взаимодействия пользователей, курсов, уроков, платежей и сессий. Подробное описание структуры и логики этих связей представлено в разделе «ER-диаграмма и набор таблиц проекта»

Я сознательно отказался от использования связей «один-к-одному» и «многие-ко-многим»: на практике они не приносят дополнительной ценности, но существенно усложняют создание и поддержку БД, особенно при наполнении тестовыми данными и масштабировании системы в будущем. Такое решение делает архитектуру более прозрачной и облегчает дальнейшую поддержку и развитие сервиса.

Примеры разработки таблиц

В данной главе я продемонстрирую часть разработанного кода. Весь код можно просмотреть в разделе: Приложении

Код для создания таблицы Users:

```
CREATE TABLE users (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  username       VARCHAR(100) NOT NULL UNIQUE,
  hash_password  VARCHAR(255) NOT NULL,
  email          VARCHAR(255) NOT NULL UNIQUE,
  balance        DECIMAL(10,2) NOT NULL DEFAULT 0.00
) ENGINE=InnoDB;
```

Код для создания таблицы Teachers:

```
CREATE TABLE teachers (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  name           VARCHAR(255) NOT NULL,
  bio            TEXT
) ENGINE=InnoDB;
```

Полный код для создания всех таблиц в Приложение 1

Пример заполнения таблиц Users и Teachers:

```
-- Заполнение Users
INSERT INTO users (username, hash_password, email) VALUES
('alice','$2y$…','alice@eduplus.com'),
('bob'  ,'$2y$…','bob@eduplus.com'),
('carol','$2y$…','carol@eduplus.com');
-- Заполнение Teachers
INSERT INTO teachers (name,bio) VALUES
('Иван Иванов','Эксперт по Python'),
('Пётр Петров','Веб-разработка и JavaScript');
```

Полный код в Приложении 2

Реализация бизнес-процессов на уровне СУБД

Классические запросы типизированные запросы состоят из следующих операторов

- SELECT — извлечение отдельных полей или всех записей из таблицы/представления;

- FROM — указание источника данных (таблица или VIEW);

- WHERE — фильтрация по условиям (например, `WHERE balance >= price` для отбора оплаченных курсов);

- GROUP BY — группировка результатов по столбцу (например, `GROUP BY course_id`);

- HAVING — фильтрация сгруппированного набора (например, только курсы с более чем 10 студентами);

- ORDER BY — сортировка по одному или нескольким полям (например, по дате `ORDER BY paid_at DESC`).

Типовые запросы представляют из себя:

```
-- 1) Список всех курсов с именем преподавателя
SELECT c.id, c.title, t.name
FROM courses c
JOIN teachers t ON c.teacher_id = t.id;

-- 2) Прогресс Alice по курсу 1
SELECT fn_user_progress(1,1) AS pct_done;

-- 3) Уроки курса 2
SELECT id, title, duration_min
FROM lessons
WHERE course_id = 2;

-- 4) Статистика по зачислениям
SELECT course_id, COUNT(*) AS students
FROM enrollments
GROUP BY course_id;

-- 5) Текущие активные сессии Alice
SELECT * FROM sessions
WHERE user_id = 1 AND completed_at IS NULL;
```

Триггер — это пользовательская процедура, автоматически выполняемая при INSERT/UPDATE/DELETE.

В EduPlus есть trg_after_payment (AFTER INSERT ON payments) – после оплаты записывает студента в enrollments и списывает баланс.

```
DELIMITER $$
CREATE TRIGGER trg_after_payment
AFTER INSERT ON payments
FOR EACH ROW
BEGIN
  INSERT IGNORE INTO enrollments(user_id, course_id)
    VALUES (NEW.user_id, NEW.course_id);

  UPDATE users
    SET balance = balance - NEW.amount
  WHERE id = NEW.user_id;
END$$
DELIMITER ;
```

Представление реализует отображение всех курсов преподавателей, зачисленных и проценты прохождения

teacher_id	teacher_name	teacher_bio	course_id	course_title	course_description	enrolled_count	total_lessons	total_revenue
1	Иван Иванов	Эксперт по Python	1	Python для начинающих	Основы Python	1	5	1000.00
1	Иван Иванов	Эксперт по Python	3	Python для начинающих	Основы синтаксиса, работа с библиотеками,...	0	2	0.00
1	Иван Иванов	Эксперт по Python	4	Продвинутый Python	Глубокое погружение: ООР, асинхронность,...	0	2	0.00
2	Пётр Петров	Веб-разработка и JavaScript	2	Frontend: HTML/CSS/JS	С нуля до PRO	0	4	0.00
2	Пётр Петров	Веб-разработка и JavaScript	5	Web-разработка	HTML, CSS, JavaScript, фреймворки React и Vue	0	2	0.00
3	Иван Иванов	Эксперт по Python и Data Science	6	SQL и базы данных	Проектирование схем, оптимизация запросо...	0	0	0.00
4	Пётр Петров	Специалист по веб-разработке и JavaScript	7	DevOps практики	CI/CD, Docker, Kubernetes, мониторинг	0	0	0.00
5	Мария Смирнова	Преподаватель по базам данных и SQL	NULL	NULL	NULL	0	0	0.00
6	Елена Кузнецова	Инструктор по DevOps и CI/CD	NULL	NULL	NULL	0	0	0.00

Рисунок 9. – Пример вывода View

Некоторый учителя просто числятся в базе данных без назначения, при желании можно выдать им назначения и курсы.

Транзакция объединяет несколько операций в атомарный блок - В sp_enroll_and_pay проверяются баланс, вставляется запись в payments, триггером создаётся enrollment. Все в START TRANSACTION…COMMIT/ROLLBACK. (Приложение 3)

Хранимые процедуры и функции (Полный код см в приложении 3):

- sp_enroll_and_pay — процедура для оплаты и зачисления.
- fn_user_progress — функция для расчёта процента выполнения курса.

Система ролей:

```
-- Администратор
CREATE USER IF NOT EXISTS 'admin'@'%'    IDENTIFIED BY 'admin_pwd';
GRANT ALL PRIVILEGES ON eduplus.* TO 'admin'@'%';

-- Студент
CREATE USER IF NOT EXISTS 'student'@'%'  IDENTIFIED BY 'stud_pwd';
GRANT SELECT, INSERT ON eduplus.sessions    TO 'student'@'%';
GRANT SELECT, INSERT ON eduplus.payments    TO 'student'@'%';
GRANT SELECT           ON eduplus.courses     TO 'student'@'%';
GRANT SELECT           ON eduplus.lessons     TO 'student'@'%';
GRANT SELECT           ON eduplus.enrollments TO 'student'@'%';

-- Преподаватель
CREATE USER IF NOT EXISTS 'teacher'@'%'  IDENTIFIED BY 'teach_pwd';
GRANT SELECT, INSERT, UPDATE ON eduplus.lessons TO 'teacher'@'%';
GRANT SELECT, UPDATE          ON eduplus.courses TO 'teacher'@'%';
```

Заключение

В ходе выполнения курсовой работы была разработана реляционная база данных для онлайн-платформы «EduPlus», полностью удовлетворяющая поставленным функциональным и нефункциональным требованиям. В результате проделанной работы:

1. Проведён анализ бизнес- и образовательных процессов дистанционного обучения и сформулированы требования к системе хранения данных.

2. Спроектирована концептуальная ER-модель и выполнено её преобразование в физическую схему баз данных MySQL, включающую семь основных таблиц (users, teachers, courses, lessons, enrollments, payments, sessions) с корректно настроенными внешними ключами.

3. Реализован полный набор механизмов СУБД:

- Типовые запросы для выборки курсов, уроков, статистики зачислений и прогресса;

- Триггер trg_after_payment для автоматического зачисления и обновления баланса;

- Представление vw_course_overview (и дополнительное vw_teacher_courses) для получения консолидированной информации о курсах и преподавателях;

- Хранимая процедура sp_enroll_and_pay и функция fn_user_progress с транзакцией и обработчиком ошибок;

- Роли (admin, student, teacher) с разграничением прав на чтение и изменение данных.

4. Наполнены таблицы тестовыми данными, проверена корректность всех операций и отсутствие конфликтов при повторном выполнении скриптов.

Полученная база данных обеспечивает целостность и консистентность информации, поддерживает ключевые образовательные сценарии: регистрацию пользователей, оплату курсов, ведение прогресса и аналитику. Архитектура легко расширяется — для внедрения новых сущностей (сертификатов, тестов, чатов) достаточно добавить таблицы и соответствующие связи. Конечно данная база

данных открыта для будущих улучшений функционала. Таким образом, разработанная база данных создаёт надёжную и масштабируемую основу для эффективного функционирования онлайн-платформы дистанционного обучения «EduPlus».

Список литературы

1. Мельник, М. В. Использование онлайн-обучения в условиях цифровизации образования / М. В. Мельник // Современные проблемы науки и образования. — 2020. — № 6. — URL: https://science-education.ru/article/view?id=30871 (дата обращения: 18.06.2025).

2. Дистанционное обучение // Википедия: свободная энциклопедия. — URL: https://ru.wikipedia.org/wiki/Дистанционное_обучение (дата обращения: 18.06.2025).

3. Бизнес-процесс // Википедия: свободная энциклопедия. — URL: https://ru.wikipedia.org/wiki/Бизнес-процесс (дата обращения: 19.06.2025

4. ER-модель // Википедия: свободная энциклопедия. — URL: https://ru.wikipedia.org/wiki/ER-модель (дата обращения: 23.06.2025).

5. MySQL 8.0 Reference Manual / MySQL Documentation. — URL: https://dev.mysql.com/doc/ (дата обращения: 24.06.2025).

6. Онлайн-платформы для обучения: особенности, функционал, востребованность // Skillspace. — URL: https://skillspace.ru/blog/onlain-platformy-dlia-obuchieniia-osobiennosti-funktsional-vostriebovannost (дата обращения: 18.06.2025).

Приложение 1.

```
-- 2. Таблица пользователей (студентов)
CREATE TABLE users (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  username       VARCHAR(100) NOT NULL UNIQUE,
  hash_password  VARCHAR(255) NOT NULL,
  email          VARCHAR(255) NOT NULL UNIQUE,
  balance        DECIMAL(10,2) NOT NULL DEFAULT 0.00
) ENGINE=InnoDB;

-- 3. Таблица преподавателей
CREATE TABLE teachers (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  name           VARCHAR(255) NOT NULL,
  bio            TEXT
) ENGINE=InnoDB;

-- 4. Таблица курсов
CREATE TABLE courses (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  title          VARCHAR(255) NOT NULL,
  description    TEXT,
  price          DECIMAL(10,2) NOT NULL,
  teacher_id     INT NOT NULL,
  FOREIGN KEY (teacher_id) REFERENCES teachers(id)
) ENGINE=InnoDB;

-- 5. Таблица уроков
CREATE TABLE lessons (
  id             INT AUTO_INCREMENT PRIMARY KEY,
  course_id      INT NOT NULL,
  title          VARCHAR(255) NOT NULL,
  content        TEXT,
  duration_min   INT NOT NULL,
  FOREIGN KEY (course_id) REFERENCES courses(id)
) ENGINE=InnoDB;
```

```
    -- 6. Таблица зачислений: enroll_date автоматически ставится
CURRENT_TIMESTAMP
    CREATE TABLE enrollments (
      id             INT AUTO_INCREMENT PRIMARY KEY,
      user_id        INT NOT NULL,
      course_id      INT NOT NULL,
      enroll_date    DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
      FOREIGN KEY (user_id)   REFERENCES users(id),
      FOREIGN KEY (course_id) REFERENCES courses(id)
    ) ENGINE=InnoDB;

    -- 7. Таблица платежей
    CREATE TABLE payments (
      id             INT AUTO_INCREMENT PRIMARY KEY,
      user_id        INT NOT NULL,
      course_id      INT NOT NULL,
      amount         DECIMAL(10,2) NOT NULL,
      paid_at        DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
      FOREIGN KEY (user_id)   REFERENCES users(id),
      FOREIGN KEY (course_id) REFERENCES courses(id)
    ) ENGINE=InnoDB;

    -- 8. Таблица сессий (просмотр урока)
    CREATE TABLE sessions (
      id             INT AUTO_INCREMENT PRIMARY KEY,
      user_id        INT NOT NULL,
      lesson_id      INT NOT NULL,
      started_at     DATETIME NOT NULL DEFAULT CURRENT_TIMESTAMP,
      completed_at   DATETIME NULL,
      FOREIGN KEY (user_id)   REFERENCES users(id),
      FOREIGN KEY (lesson_id) REFERENCES lessons(id)
) ENGINE=InnoDB;
```

Приложение 2.

```
-- 14. Пример наполнения данными
INSERT INTO users (username, hash_password, email) VALUES
('alice','$2y$…','alice@eduplus.com'),
('bob'  ,'$2y$…','bob@eduplus.com'),
('carol','$2y$…','carol@eduplus.com');

INSERT INTO teachers (name,bio) VALUES
('Иван Иванов','Эксперт по Python'),
('Пётр Петров','Веб-разработка и JavaScript');

INSERT INTO courses (title,description,price,teacher_id) VALUES
('Python для начинающих','Основы Python',500.00,1),
('Frontend: HTML/CSS/JS','С нуля до PRO',700.00,2);

INSERT INTO lessons (course_id,title,content,duration_min) VALUES
(1,'Введение','Что такое Python',10),
(1,'Переменные','Типы и переменные',20),
(2,'HTML основы','Разметка страницы',15),
(2,'CSS стили','Оформление',25);
INSERT IGNORE INTO users (username, hash_password, email, balance) VALUES
  ('alice', '$2y$abc1', 'alice@eduplus.com', 1000.00),
  ('bob',   '$2y$abc2', 'bob@eduplus.com',   800.00),
  ('carol', '$2y$abc3', 'carol@eduplus.com', 1200.00),
  ('dave',  '$2y$abc4', 'dave@eduplus.com',   500.00),
  ('erin',  '$2y$abc5', 'erin@eduplus.com',  1500.00),
  ('frank', '$2y$abc6', 'frank@eduplus.com',  700.00);

-- Преподаватели
INSERT IGNORE INTO teachers (name, bio) VALUES
  ('Иван Иванов',     'Эксперт по Python и Data Science'),
  ('Пётр Петров',     'Специалист по веб-разработке и JavaScript'),
  ('Мария Смирнова',  'Преподаватель по базам данных и SQL'),
  ('Елена Кузнецова', 'Инструктор по DevOps и CI/CD');

-- Курсы
INSERT IGNORE INTO courses (title, description, price, teacher_id) VALUES
```

```
  ('Python для начинающих', 'Основы синтаксиса, работа с библиотеками, первые проекты', 500.00, 1),
  ('Продвинутый Python', 'Глубокое погружение: ООР, асинхронность, оптимизация', 700.00, 1),
  ('Web-разработка', 'HTML, CSS, JavaScript, фреймворки React и Vue', 800.00, 2),
  ('SQL и базы данных', 'Проектирование схем, оптимизация запросов, транзакции', 600.00, 3),
  ('DevOps практики', 'CI/CD, Docker, Kubernetes, мониторинг', 900.00, 4)
ON DUPLICATE KEY UPDATE title=VALUES(title);

-- Уроки
INSERT IGNORE INTO lessons (course_id, title, content, duration_min) VALUES
  (1, 'Введение в Python', 'Обзор языка, установка окружения', 15),
  (1, 'Переменные и типы данных', 'int, float, str, list, dict', 20),
  (1, 'Условные операторы', 'if, else, elif', 25),
  (2, 'Классы и объекты', 'Создание классов, наследование', 30),
  (2, 'Асинхронность', 'asyncio, задачики, event loop', 35),
  (3, 'Основы HTML/CSS', 'Теги, стили, макеты', 20),
  (3, 'JavaScript для веба', 'DOM, события, ES6+', 25),
  (4, 'Введение в SQL', 'SELECT, INSERT, UPDATE, DELETE', 15),
  (4, 'Продвинутые JOIN', 'INNER, LEFT, RIGHT, FULL JOIN', 20),
  (5, 'Введение в Docker', 'Контейнеризация приложений', 30),
  (5, 'Kubernetes в практике', 'Создание подов, сервисов, deployment', 40)
ON DUPLICATE KEY UPDATE title=VALUES(title);
```

Приложение 3.

```
-- Студент
CREATE USER IF NOT EXISTS 'student'@'%'  IDENTIFIED BY 'stud_pwd';
GRANT SELECT, INSERT ON eduplus.sessions    TO 'student'@'%';
GRANT SELECT, INSERT ON eduplus.payments    TO 'student'@'%';
GRANT SELECT           ON eduplus.courses     TO 'student'@'%';
GRANT SELECT           ON eduplus.lessons     TO 'student'@'%';
GRANT SELECT           ON eduplus.enrollments TO 'student'@'%';

-- Преподаватель
CREATE USER IF NOT EXISTS 'teacher'@'%'  IDENTIFIED BY 'teach_pwd';
GRANT SELECT, INSERT, UPDATE ON eduplus.lessons TO 'teacher'@'%';
GRANT SELECT, UPDATE          ON eduplus.courses TO 'teacher'@'%';

-- 10. Пользовательская функция: прогресс в курсе
DELIMITER $$
CREATE FUNCTION fn_user_progress(u_id INT, c_id INT)
RETURNS DECIMAL(5,2) DETERMINISTIC
BEGIN
  DECLARE total_lessons INT;
  DECLARE done_lessons  INT;
  DECLARE pct           DECIMAL(5,2);

  SELECT COUNT(*) INTO total_lessons
    FROM lessons WHERE course_id = c_id;

  SELECT COUNT(*) INTO done_lessons
    FROM sessions s
    JOIN lessons l ON s.lesson_id = l.id
    WHERE s.user_id = u_id
      AND l.course_id = c_id
      AND s.completed_at IS NOT NULL;

  IF total_lessons = 0 THEN
    SET pct = 0;
  ELSE
    SET pct = done_lessons/total_lessons*100;
```

```
  END IF;

  RETURN pct;
END$$
DELIMITER ;

-- 11. Триггер: после оплаты автоматически зачисляем и списываем баланс
DELIMITER $$
CREATE TRIGGER trg_after_payment
AFTER INSERT ON payments
FOR EACH ROW
BEGIN
  INSERT IGNORE INTO enrollments(user_id, course_id)
    VALUES (NEW.user_id, NEW.course_id);

  UPDATE users
    SET balance = balance - NEW.amount
  WHERE id = NEW.user_id;
END$$
DELIMITER ;

-- Вьюшки
CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `vw_course_overview` AS
    SELECT
        `c`.`id` AS `course_id`,
        `c`.`title` AS `course_title`,
        `c`.`description` AS `course_description`,
        `t`.`name` AS `teacher`,
        COUNT(DISTINCT `e`.`user_id`) AS `enrolled_count`,
        COUNT(DISTINCT `l`.`id`) AS `total_lessons`,
        IFNULL(SUM(`p`.`amount`), 0) AS `total_revenue`,
        IFNULL(AVG(FN_USER_PROGRESS(`e`.`user_id`, `c`.`id`)),
                0) AS `avg_progress_pct`
    FROM
        ((((`courses` `c`
```

```
        JOIN `teachers` `t` ON ((`c`.`teacher_id` = `t`.`id`)))
        LEFT JOIN `enrollments` `e` ON ((`c`.`id` = `e`.`course_id`)))
        LEFT JOIN `lessons` `l` ON ((`c`.`id` = `l`.`course_id`)))
        LEFT JOIN `payments` `p` ON ((`c`.`id` = `p`.`course_id`)))
    GROUP BY `c`.`id`

CREATE
    ALGORITHM = UNDEFINED
    DEFINER = `root`@`localhost`
    SQL SECURITY DEFINER
VIEW `vw_course_overview` AS
    SELECT
        `c`.`id` AS `course_id`,
        `c`.`title` AS `course_title`,
        `c`.`description` AS `course_description`,
        `t`.`name` AS `teacher`,
        COUNT(DISTINCT `e`.`user_id`) AS `enrolled_count`,
        COUNT(DISTINCT `l`.`id`) AS `total_lessons`,
        IFNULL(SUM(`p`.`amount`), 0) AS `total_revenue`,
        IFNULL(AVG(FN_USER_PROGRESS(`e`.`user_id`, `c`.`id`)),
                0) AS `avg_progress_pct`
    FROM
        ((((`courses` `c`
        JOIN `teachers` `t` ON ((`c`.`teacher_id` = `t`.`id`)))
        LEFT JOIN `enrollments` `e` ON ((`c`.`id` = `e`.`course_id`)))
        LEFT JOIN `lessons` `l` ON ((`c`.`id` = `l`.`course_id`)))
        LEFT JOIN `payments` `p` ON ((`c`.`id` = `p`.`course_id`)))
    GROUP BY `c`.`id`

-- хранимая процедура с обработчиком ошибок
CREATE DEFINER=`root`@`localhost` PROCEDURE `sp_enroll_and_pay`(
  IN p_user   INT,
  IN p_course INT
)
BEGIN
  DECLARE course_price DECIMAL(10,2);
  DECLARE user_bal     DECIMAL(10,2);

  DECLARE EXIT HANDLER FOR SQLEXCEPTION
```

```
  BEGIN
    ROLLBACK;
    SELECT 'Ошибка: транзакция отменена.' AS msg;
  END;

  START TRANSACTION;
    SELECT price INTO course_price
      FROM courses
      WHERE id = p_course
      FOR UPDATE;

    SELECT balance INTO user_bal
      FROM users
      WHERE id = p_user
      FOR UPDATE;

    IF user_bal >= course_price THEN
      INSERT INTO payments(user_id, course_id, amount)
        VALUES (p_user, p_course, course_price);
      -- enrollments и списание баланса сделает триггер
      COMMIT;
      SELECT 'Успешно зачислен на курс!' AS msg;
    ELSE
      ROLLBACK;
      SELECT 'Недостаточно средств.' AS msg;
    END IF;
END
```

Приложение 4.

```
-- 1) Список всех курсов с именем преподавателя
SELECT c.id, c.title, t.name
FROM courses c
JOIN teachers t ON c.teacher_id = t.id;

-- 2) Прогресс Alice по курсу 1
SELECT fn_user_progress(1,1) AS pct_done;

-- 3) Уроки курса 2
SELECT id, title, duration_min
FROM lessons
WHERE course_id = 2;

-- 4) Статистика по зачислениям
SELECT course_id, COUNT(*) AS students
FROM enrollments
GROUP BY course_id;

-- 5) Текущие активные сессии Alice
SELECT * FROM sessions
WHERE user_id = 1 AND completed_at IS NULL;
```