АКТ (ф) СПбГУТ

ОТЧЁТ

ПО ПРАКТИЧЕСКИМ РАБОТАМ

Студент: Ларионов

Преподаватель: Гвоздёв

Архангельск 2025

Практическая работа №2

Ответы на контрольные вопросы:

1. MySQL Workbench — это интегрированная среда разработки, которая используется для управления базами данных MySQL. Она предоставляет инструменты для проектирования, моделирования, администрирования и разработки баз данных.  
   1. Откройте MySQL Workbench.
   2. Выберите "File" (Файл) > "New Model" (Новая модель).
   3. В открывшемся окне выберите "Add Diagram" (Добавить диаграмму).
   4. Сущности (Entities): представляют объекты, которые имеют значения и свойства.
   5. Атрибуты (Attributes): характеристики сущностей.
   6. Связи (Relationships): показывают, как сущности связаны друг с другом.
   7. Наследование (Inheritance): позволяет создавать подтипы сущностей.
   8. Индексы (Indexes): используются для оптимизации запросов.
   9. Откройте таблицу, для которой хотите создать индекс.
   10. Перейдите на вкладку "Indexes" (Индексы).
   11. Нажмите на кнопку "Add Index" (Добавить индекс).
   12. Введите имя индекса и выберите тип индекса (например, уникальный или обычный).
   13. Выберите столбцы, которые будут частью индекса.
   14. Нажмите "Apply" (Применить), чтобы сохранить изменения.
   15. Обычные индексы (Regular Index): стандартные индексы, которые ускоряют поиск.
   16. Уникальные индексы (Unique Index): гарантируют уникальность значений в столбце.
   17. Полнотекстовые индексы (Full-text Index): используются для полнотекстового поиска в текстовых полях.
   18. Индексы по нескольким столбцам (Composite Index): индексы, которые включают несколько столбцов, что позволяет ускорить поиск по комбинации значений.

Практическая работа №3

* 1. Проектирования и моделирования баз данных.
  2. Выполнения SQL-запросов и управления данными.
  3. Администрирования серверов MySQL, включая управление пользователями и правами доступа.
  4. Создания и редактирования схем баз данных и визуализации их структуры.
  5. Выполнения резервного копирования и восстановления данных.

1. Прямой инжиниринг — это процесс, при котором существующая база данных используется для создания ее визуальной модели и структуры в MySQL Workbench. Это позволяет пользователям анализировать и документировать уже существующие схемы баз данных, а также вносить изменения в их структуру на основе визуального представления.  
   1. Откройте MySQL Workbench.
   2. Перейдите в раздел "Database" (База данных) в верхнем меню.
   3. Выберите "Reverse Engineer..." (Обратный инжиниринг...).
   4. Введите данные для подключения к серверу MySQL, если это необходимо.
   5. Выберите базу данных, которую хотите проанализировать, и нажмите "Next" (Далее).
   6. Выберите объекты (таблицы, представления, процедуры и т.д.), которые хотите импортировать в модель, и нажмите "Execute" (Выполнить).
   7. После завершения процесса вы получите визуальное представление структуры базы данных.
2. Определения таблиц (CREATE TABLE) с указанием их структуры, типов данных и атрибутов.
3. Связи между таблицами (FOREIGN KEY), показывающие, как они взаимодействуют друг с другом.
4. Индексы, уникальные ключи и другие ограничения, которые применяются к столбцам.
5. Определения представлений (VIEW), триггеров (TRIGGER) и хранимых процедур (STORED PROCEDURE), если они существуют в базе данных.

Практическая работа №4

1. SELECT [столбцы]

FROM [таблицы]

WHERE [условия]

GROUP BY [столбцы]

HAVING [условия]

ORDER BY [столбцы];

1. После ключевого слова SELECT указываются столбцы, которые вы хотите выбрать из таблицы. Это могут быть конкретные столбцы (например, column1, column2) или символ \*, который обозначает выбор всех столбцов из таблицы (например, SELECT \*)
2. После ключевого слова FROM указывается имя таблицы или таблиц, из которых будут извлекаться данные. Если данные выбираются из нескольких таблиц, может использоваться оператор JOIN для их объединения.
3. После ключевого слова WHERE указываются условия фильтрации данных. Эти условия определяют, какие записи будут включены в результат запроса (например, WHERE column1 = 'значение').  
   1. WHERE: используется для фильтрации строк до выполнения группировки. Применяется к отдельным строкам.
   2. HAVING: используется для фильтрации групп после выполнения группировки. Применяется к агрегированным данным (например, после использования функций COUNT, SUM и т.д.)
4. После ключевого слова GROUP BY указываются столбцы, по которым будет происходить группировка данных. Это позволяет агрегировать данные по указанным столбцам (например, GROUP BY column1).  
   1. COUNT(): подсчет количества строк.
   2. SUM(): сумма значений столбца.
   3. AVG(): среднее значение столбца.
   4. MAX(): максимальное значение.
   5. MIN(): минимальное значение.
   6. Эти функции требуют группировки данных для получения агрегированных результатов.
5. После ключевого слова ORDER BY указываются столбцы, по которым будет происходить сортировка результатов запроса. Можно указать направление сортировки: ASC (по возрастанию) или DESC (по убыванию). Например, ORDER BY column1 ASC.
6. SELECT column1, column2, (column1 + column2) AS total

FROM table\_name;

Практическая работа №5

1. После ключевого слова FROM указывается имя таблицы или таблиц, из которых будут извлекаться данные. Если данные выбираются из нескольких таблиц, может использоваться оператор JOIN для их объединения.
2. SELECT столбцы

FROM таблица1

JOIN таблица2 ON условие;

* 1. INNER JOIN: возвращает только те строки, которые имеют совпадения в обеих таблицах.
  2. LEFT JOIN (или LEFT OUTER JOIN): возвращает все строки из левой таблицы и совпадающие строки из правой таблицы. Если нет совпадений, в правой таблице будут NULL.
  3. RIGHT JOIN (или RIGHT OUTER JOIN): возвращает все строки из правой таблицы и совпадающие строки из левой таблицы. Если нет совпадений, в левой таблице будут NULL.
  4. FULL JOIN (или FULL OUTER JOIN): возвращает все строки из обеих таблиц, с NULL в местах, где нет совпадений.
  5. CROSS JOIN: возвращает декартово произведение двух таблиц, то есть все возможные комбинации строк из обеих таблиц.
  6. LEFT OUTER JOIN: возвращает все строки из левой таблицы и совпадающие строки из правой таблицы. Если совпадений нет, в результатах будут NULL для столбцов правой таблицы.
  7. RIGHT OUTER JOIN: возвращает все строки из правой таблицы и совпадающие строки из левой таблицы. Если совпадений нет, в результатах будут NULL для столбцов левой таблицы.

1. Подзапрос — это запрос, вложенный в другой SQL-запрос. Он может использоваться для фильтрации данных, вычисления значений или получения данных, которые затем используются в основном запросе. Подзапросы могут быть использованы в условиях WHERE, FROM или SELECT.
2. Ключевое слово UNION используется для объединения результатов двух или более SELECT-запросов в один набор результатов. Все запросы должны иметь одинаковое количество столбцов с совместимыми типами данных.

Практическая работа №6

* 1. Неправильные типы данных: попытка вставить данные, которые не соответствуют типу столбца (например, вставка строки в числовое поле).
  2. Отсутствие обязательных полей: если в таблице есть столбцы без значения по умолчанию, и они не указаны в запросе, это приведет к ошибке.
  3. Нарушение ограничений уникальности: попытка вставить дубликаты в столбец, который имеет уникальное ограничение.
  4. Ошибки в синтаксисе: неправильный синтаксис SQL-запроса, например, пропущенные запятые или скобки.
  5. Проблемы с внешними ключами: попытка вставить запись, которая ссылается на несуществующую запись в другой таблице.

1. В запросе на обновление после ключевого слова SET указываются обновляемые поля.
2. В одном запросе на обновление можно обновлять несколько полей. Количество обновляемых полей ограничено только синтаксисом SQL и размером строки данных.
3. В стандартном SQL можно обновлять только одну таблицу за один запрос на обновление. Однако можно использовать подзапросы для обновления значений, основанных на данных из других таблиц.
4. DELETE FROM orders

WHERE customer\_id = (SELECT id FROM customers WHERE condition);

1. Для группового ввода записей из одной таблицы в другую обе таблицы должны иметь совместимые структуры. Это означает, что:

* 1. Количество столбцов в обеих таблицах должно совпадать.
  2. Типы данных соответствующих столбцов должны быть совместимыми.
  3. Если в целевой таблице есть обязательные поля, они должны быть заполнены значениями, либо должны иметь значения по умолчанию.

Практическая работа №7

1. Преимущества применения встроенных функций в MySQL включают:
   1. Упрощение запросов: Встроенные функции позволяют выполнять сложные вычисления и преобразования данных без необходимости писать многострочный код.
   2. Увеличение производительности: Они оптимизированы для работы с базами данных и могут выполнять операции быстрее, чем эквивалентный код на SQL.
   3. Снижение ошибок: Использование проверенных функций уменьшает вероятность ошибок в логике запросов.
   4. Удобство и читаемость: Код становится более понятным и легче читается, так как функции выполняют конкретные задачи.
2. В MySQL есть несколько видов системных функций, включая:
   1. Агрегатные функции: Например, COUNT(), SUM(), AVG(), MAX(), MIN(), которые обрабатывают наборы данных и возвращают одно значение.
   2. Строковые функции: Например, CONCAT(), SUBSTRING(), LENGTH(), которые работают со строками.
   3. Числовые функции: Например, ROUND(), FLOOR(), CEIL(), которые выполняют математические операции.
   4. Дата и время функции: Например, NOW(), CURDATE(), DATE\_FORMAT(), которые работают с датами и временем.
   5. Условные функции: Например, IF(), CASE, которые позволяют выполнять условия в запросах.
3. Передаваемые в функцию параметры указываются в круглых скобках после названия функции. Параметры могут быть разделены запятыми. SELECT CONCAT('Hello, ', 'World!');
4. Встроенную функцию можно вызвать, указав ее имя и передав необходимые параметры в круглых скобках.

Практическая работа №8

1. Полнотекстовый поиск — это метод поиска, который позволяет находить записи в текстовых полях базы данных, основываясь на содержимом текста, а не только на точных совпадениях. Он позволяет искать слова и фразы, игнорируя при этом незначительные слова (например, предлоги) и учитывая форму слов, что делает поиск более гибким и эффективным.
2. Команда MATCH() используется для выполнения полнотекстового поиска по текстовым полям. Она принимает в качестве параметров одно или несколько полей, по которым будет производиться поиск.
3. Команда AGAINST() используется совместно с MATCH() для определения поискового запроса. Она принимает строку для поиска и может использоваться для указания различных режимов поиска (например, естественный режим, Boolean режим).
4. Чтобы включить логический режим в процессе полнотекстового поиска, необходимо указать IN BOOLEAN MODE в команде AGAINST().
5. Полнотекстовый индекс можно создать с помощью команды CREATE INDEX.
6. Полнотекстовый индекс требуется создавать, когда необходимо выполнять полнотекстовый поиск по текстовым полям, особенно в случаях, когда:
   1. Поля содержат большие объемы текста.
   2. Необходимо быстро находить записи по ключевым словам или фразам.
   3. Требуется улучшить производительность поиска по текстовым данным.
7. Значение релевантности в процессе полнотекстового поиска показывает, насколько найденные записи соответствуют запросу пользователя. Чем выше значение релевантности, тем более релевантна запись к поисковому запросу. Это значение рассчитывается на основе различных факторов, таких как частота появления искомых слов в тексте и их расположение.

Практическая работа №9

1. Транзакция — это последовательность операций, которые выполняются как единое целое. Транзакция должна удовлетворять четырем основным свойствам, известным как ACID.
2. Откат транзакции в MySQL выполняется с помощью команды ROLLBACK. Эта команда отменяет все изменения, сделанные в текущей транзакции, и восстанавливает состояние базы данных до начала транзакции.
3. Фиксация транзакции выполняется с помощью команды COMMIT. Эта команда сохраняет все изменения, сделанные в текущей транзакции, и делает их постоянными.
4. Точка сохранения (SAVEPOINT) позволяет установить промежуточный уровень, к которому можно откатить транзакцию. Это полезно, если нужно отменить часть операций, но оставить другие.

Практическая работа №10

1. Событие в MySQL — это объект базы данных, который позволяет планировать выполнение заданных операций (например, SQL-запросов) автоматически в определенное время или с заданной периодичностью. События полезны для автоматизации задач, таких как резервное копирование данных, обновление статистики или очистка старых записей.
2. CREATE EVENT имя\_события

ON SCHEDULE расписание

DO

SQL\_операция;

* 1. SHOW EVENTS;
  2. SELECT \* FROM information\_schema.EVENTS WHERE EVENT\_SCHEMA = 'имя\_вашей\_базы\_данных';

1. Периодичность выполнения события задается с помощью конструкции ON SCHEDULE, где можно указать, как часто должно выполняться событие.

Практическая работа №11

1. В инструкции CREATE TABLE для определения первичного ключа используется предложение PRIMARY KEY.
2. Для определения внешнего ключа в инструкции CREATE TABLE используется предложение FOREIGN KEY.
3. Ограничения для значений столбца задаются с помощью различных ключевых слов, таких как NOT NULL, UNIQUE, CHECK, DEFAULT и других.  
   1. На столбец или таблицу ссылается внешний ключ в другой таблице.
   2. Столбец является частью первичного ключа или уникального ограничения.
   3. Таблица содержит данные, и вы не используете команду DROP TABLE с параметром CASCADE, если есть зависимости.
4. Для создания совместно уникального ограничения на пару столбцов используется предложение UNIQUE.
5. Команда DROP TABLE используется для удаления таблицы из базы данных. При этом удаляются все данные, структура таблицы и связанные с ней индексы и ограничения.
6. Индекс — это структура данных, которая улучшает скорость операций выборки (SELECT) в таблице. Индексы позволяют быстро находить строки по значениям одного или нескольких столбцов.
7. Индексы предназначены для ускорения поиска и сортировки данных в базе данных. Они значительно снижают время выполнения запросов, особенно при работе с большими объемами данных.  
   1. Таблицах (столбцы таблиц).
   2. View (представлениях).
   3. Полях, участвующих в уникальных ограничениях и первичных ключах.

Практическая работа №12

1. MySQL Workbench — это интегрированная среда разработки, которая используется для управления базами данных MySQL. Она предоставляет инструменты для проектирования, моделирования, администрирования и разработки баз данных.
2. Обратный инжиниринг — это процесс создания модели базы данных на основе существующей базы данных. Это позволяет извлечь структуру, связи и другие характеристики базы данных, чтобы лучше понять её архитектуру или для дальнейшего редизайна. В контексте баз данных это может включать в себя создание диаграмм сущностей и связей (ER-диаграмм) на основе существующих таблиц и их взаимосвязей.
3. 1. Запустите MySQL Workbench и подключитесь к вашему серверу MySQL.
   2. Перейдите в меню Database (База данных).
   3. Выберите Reverse Engineer (Обратный инжиниринг).
   4. В открывшемся окне выберите соединение с сервером, на котором находится ваша база данных, и нажмите Next (Далее).
   5. Выберите базу данных, которую хотите проанализировать, и нажмите Next.
   6. MySQL Workbench начнет извлечение схемы базы данных. После завершения вы сможете просмотреть таблицы, представления, процедуры и другие объекты.
   7. Нажмите Execute (Выполнить), чтобы создать ER-диаграмму на основе извлеченной информации.
   8. Вы сможете сохранить или экспортировать созданную диаграмму для дальнейшего использования.

Практическая работа №13

1. Представление (или view) — это виртуальная таблица, которая формируется на основе результата SQL-запроса. В отличие от обычных таблиц, представления не хранят данные физически; они содержат только запрос, который извлекает данные из одной или нескольких таблиц. Это позволяет пользователям работать с данными более удобным способом, не изменяя исходные таблицы.  
   1. Упрощение сложных запросов: Пользователи могут обращаться к представлениям вместо написания сложных SQL-запросов каждый раз.
   2. Безопасность: Можно ограничить доступ к определённым столбцам или строкам таблицы, предоставляя доступ только к представлениям.
   3. Агрегация данных: Представления могут использоваться для агрегирования данных и представления их в более удобном формате.
   4. Сохранение бизнес-логики: Позволяют инкапсулировать бизнес-правила и логику в SQL-запросах.
2. Актуальность содержимого представления достигается за счет того, что оно всегда отображает данные, актуальные на момент выполнения запроса. При каждом обращении к представлению выполняется SQL-запрос, который извлекает свежие данные из базовых таблиц. Таким образом, изменения в базовых таблицах автоматически отражаются в представлении.
3. Да, представления могут включать информацию из нескольких таблиц. Это достигается путем написания SQL-запроса, который использует операторы JOIN, чтобы объединить данные из разных таблиц.  
   1. Они должны быть основаны на одной таблице или на нескольких таблицах с простыми соединениями.
   2. Представление не должно содержать агрегатных функций (SUM, COUNT и т. д.), группировок (GROUP BY), подзапросов или других сложных операций.
   3. Все столбцы, которые могут быть обновлены, должны быть однозначно определены.
4. Чтобы создать обновляемое представление, необходимо использовать простой SQL-запрос, который соответствует перечисленным требованиям. Пример создания обновляемого представления:  
     
   CREATE VIEW simple\_view AS

SELECT id, name

FROM my\_table;

Практическая работа №14

1. Функции пользователя в MySQL — это программные конструкции, которые позволяют выполнять определенные операции и возвращать значения. Они могут использоваться для:
   1. Повторного использования кода: Функции позволяют избежать дублирования кода, так как одна и та же функция может быть вызвана из разных мест.
   2. Упрощения сложных операций: Сложные вычисления могут быть инкапсулированы в функции, что упрощает написание SQL-запросов.
   3. Упрощения логики: Функции позволяют реализовать бизнес-логику на уровне базы данных, что может улучшить производительность и читаемость кода.
   4. Скалярные функции возвращают одно значение (например, число, строку, дату). Они принимают входные параметры и могут использоваться в выражениях, таких как SELECT, WHERE и т.д. Пример скалярной функции: LENGTH(), которая возвращает длину строки.
   5. Табличные функции возвращают набор строк, который может быть использован как таблица в SQL-запросах. Они могут быть полезны для работы с более сложными данными. Пример: функция, которая возвращает список всех заказов для конкретного клиента
2. В MySQL переменные могут быть объявлены с помощью ключевого слова SET или SELECT.
3. CREATE FUNCTION calculate\_area(radius FLOAT)

RETURNS FLOAT

BEGIN

RETURN PI() \* radius \* radius;

END;

В этом примере функция calculate\_area принимает один параметр radius и возвращает площадь круга.

Практическая работа №15

1. Хранимые процедуры в MySQL — это набор SQL-операторов, которые сохраняются в базе данных и могут быть выполнены по запросу. Они используются для:  
   1. Инкапсуляции логики: Хранимые процедуры позволяют объединить несколько SQL-запросов и бизнес-логики в одном месте, что упрощает управление кодом.
   2. Упрощения выполнения сложных операций: Они могут выполнять сложные операции с данными, такие как обновление, вставка или удаление, с минимальным количеством кода, который необходимо писать каждый раз.
   3. Повышения производительности: Поскольку хранимые процедуры компилируются и хранятся на сервере, они могут выполняться быстрее, чем если бы каждый запрос отправлялся отдельно.
   4. Безопасности: Можно ограничить доступ к данным, предоставляя пользователям возможность вызывать только определенные хранимые процедуры, а не выполнять произвольные SQL-запросы.
   5. Возвращаемое значение: Функции пользователя всегда возвращают значение и могут использоваться в выражениях. Хранимые процедуры, в свою очередь, могут не возвращать значение, а просто выполнять действия.
   6. Использование: Функции могут быть использованы в SQL-запросах (например, в SELECT), тогда как хранимые процедуры вызываются с помощью команды CALL и не могут быть использованы в выражениях.
   7. Количество параметров: Хранимые процедуры могут иметь выходные параметры, тогда как функции возвращают только одно значение.
2. Выходные параметры хранимой процедуры задаются при её объявлении с использованием ключевого слова OUT.
3. Хранимую процедуру может вызывать любой пользователь, у которого есть соответствующие привилегии на выполнение этой процедуры. Привилегии могут быть назначены с помощью команды GRANT. Если у пользователя есть необходимые права, он может вызывать хранимую процедуру с помощью команды CALL.

Практическая работа №16

1. Триггер — это специальный объект базы данных, который автоматически выполняет заданный набор действий (SQL-операторов) в ответ на определенные события, происходящие в таблице, такие как вставка, обновление или удаление данных. Триггеры помогают автоматизировать бизнес-логику и поддерживать целостность данных, выполняя действия без необходимости вручную запускать SQL-запросы.  
   1. NEW: Эта таблица содержит новые значения строк, которые будут вставлены или обновлены. Например, если вы выполняете операцию INSERT или UPDATE, вы можете обратиться к полям новой строки через NEW.column\_name.

* 1. OLD: Эта таблица содержит старые значения строк, которые были удалены или обновлены. Например, если вы выполняете операцию DELETE или UPDATE, вы можете обратиться к полям старой строки через OLD.column\_name.

1. AFTER-триггер срабатывает после выполнения определенного события (вставка, обновление или удаление) на таблице. Это означает, что действия, определенные в триггере, будут выполнены только после того, как операция изменения данных завершится успешно. Например, если вы используете AFTER-триггер для обновления другой таблицы, он будет выполнен только после того, как данные были успешно вставлены или обновлены в исходной таблице.  
   1. Поддержание целостности данных: Они могут использоваться для автоматического обновления связанных данных в других таблицах после изменения данных в основной таблице.
   2. Логирование изменений: AFTER-триггеры могут записывать изменения в отдельные таблицы для аудита или анализа.
   3. Выполнение дополнительных действий: Например, отправка уведомлений или выполнение расчетов, которые должны происходить после изменения данных.

Практическая работа №17

1. Триггер — это объект базы данных, который автоматически выполняет заданные действия (SQL-операторы) в ответ на определенные события, такие как вставка, обновление или удаление данных в таблице. Триггеры помогают автоматизировать бизнес-логику, обеспечивать целостность данных и выполнять дополнительные действия без необходимости вручную запускать SQL-запросы.  
   1. BEFORE-триггеры: Эти триггеры срабатывают перед выполнением операции (вставка, обновление или удаление). Они позволяют вам изменять данные перед их записью в базу данных или выполнять проверки, которые могут предотвратить операцию.
   2. AFTER-триггеры: Эти триггеры срабатывают после успешного выполнения операции. Они могут использоваться для выполнения действий, которые должны происходить только после изменения данных, например, логирования изменений или обновления связанных таблиц.
2. BEFORE-триггер срабатывает перед выполнением операции изменения данных (вставка, обновление или удаление) в таблице. Это позволяет вам, например, выполнять валидацию данных или изменять значения, которые будут записаны в базу данных, прежде чем они фактически будут сохранены.  
   1. Валидация данных: Вы можете проверить, соответствуют ли данные определенным критериям перед их вставкой или обновлением. Если данные не соответствуют требованиям, можно отменить операцию.
   2. Изменение данных: Вы можете изменять значения, которые будут записаны в базу данных. Например, можно автоматически преобразовать формат даты или изменить регистр строк.
   3. Предотвращение операций: Если данные не удовлетворяют каким-либо условиям, можно вызвать ошибку и предотвратить выполнение операции.

Практическая работа №18

* 1. Управления доступом: Позволяет ограничить доступ к базе данных только тем пользователям, которые имеют соответствующие права. Это помогает защитить данные от несанкционированного доступа.
  2. Аудита и отслеживания действий: Каждому пользователю можно назначить уникальные учетные данные, что позволяет отслеживать, кто и какие действия выполнял в системе.
  3. Разграничения ролей: Разные пользователи могут иметь разные уровни доступа и привилегий, что позволяет организовать работу с базой данных более эффективно и безопасно.
  4. Несанкционированный доступ: Попытки доступа к данным без соответствующих прав.
  5. Вредоносные атаки: SQL-инъекции, вирусы и другие методы, направленные на компрометацию данных.
  6. Утечка данных: Неправомерное распространение конфиденциальной информации.
  7. Потеря данных: Уничтожение или повреждение данных в результате ошибок или атак.
  8. Неавторизованные изменения: Изменения данных, произведенные пользователями без соответствующих прав.

1. Угроза, связанная с несанкционированным доступом, может быть устранена с помощью эффективного разграничения доступа. Это достигается путем создания пользователей с определенными правами и привилегиями, что позволяет контролировать, кто может выполнять какие операции в базе данных.
2. Пользователи в MySQL идентифицируются с помощью комбинации имени пользователя и хоста, с которого они подключаются. Каждому пользователю присваивается уникальное имя (например, username@hostname), что позволяет MySQL различать пользователей и назначать им различные привилегии.  
   1. Уровень аутентификации: Проверка подлинности пользователя при подключении к базе данных.
   2. Уровень авторизации: Определение прав доступа и привилегий для каждого пользователя.
   3. Уровень шифрования: Защита данных при передаче и хранении, чтобы предотвратить несанкционированный доступ.
   4. Уровень аудита и мониторинга: Отслеживание действий пользователей и изменений в базе данных для обеспечения безопасности и соблюдения норм.
   5. SELECT: Позволяет выполнять запросы на чтение данных.
   6. INSERT: Позволяет добавлять новые записи в таблицы.
   7. UPDATE: Позволяет изменять существующие записи.
   8. DELETE: Позволяет удалять записи из таблиц.
   9. CREATE: Позволяет создавать новые таблицы и базы данных.
   10. DROP: Позволяет удалять таблицы и базы данных.
   11. GRANT: Позволяет назначать привилегии другим пользователям.
   12. EXECUTE: Позволяет выполнять хранимые процедуры.

Практическая работа №19

* 1. Защиты данных: Резервные копии позволяют восстановить данные в случае их потери, повреждения или уничтожения.
  2. Восстановления после сбоев: В случае аппаратных или программных сбоев резервные копии позволяют быстро восстановить работоспособность системы.
  3. Минимизации простоев: Быстрое восстановление данных помогает уменьшить время простоя бизнеса.
  4. Аудита и восстановления: Резервные копии могут использоваться для восстановления предыдущих версий данных, что полезно для аудита и анализа.
  5. Полное резервное копирование: Создает полную копию всей базы данных, включая все таблицы и данные. Это самый надежный способ, но и самый ресурсозатратный.
  6. Потабличное резервное копирование: Создает резервные копии отдельных таблиц. Это позволяет экономить время и ресурсы, но может усложнить процесс восстановления, если требуется восстановить всю базу данных.
  7. Локальные диски: На серверах или компьютерах, где находится база данных.
  8. Сетевые хранилища: На удаленных серверах или NAS (сетевых устройствах хранения).
  9. Облачные сервисы: В облачных хранилищах, таких как Amazon S3, Google Drive или другие облачные решения.
  10. Внешние носители: На USB-накопителях или внешних жестких дисках для дополнительной безопасности.
  11. Объема изменений данных: Чем чаще данные обновляются, тем чаще следует выполнять резервное копирование.
  12. Критичности данных: Для критически важных данных резервные копии могут выполняться ежедневно или даже несколько раз в день.
  13. Политик компании: Некоторые организации устанавливают свои собственные правила и графики резервного копирования.
  14. mysqldump: Утилита командной строки для создания текстовых файлов с SQL-запросами для восстановления базы данных.
  15. MySQL Enterprise Backup: Коммерческое решение для создания резервных копий с поддержкой инкрементального и дифференциального резервного копирования.
  16. Файловые системы: Копирование файлов данных MySQL (например, с помощью rsync или cp) при остановленной базе данных.
  17. Логирование изменений: Использование бинарных логов для создания резервных копий, которые можно использовать для восстановления данных до определенного момента времени.
  18. С использованием mysqldump: Запустите команду:

mysql -u username -p database\_name < backup\_file.sql

Это восстановит базу данных из SQL-файла.

* 1. С использованием MySQL Enterprise Backup: Следуйте инструкциям для восстановления, предоставленным в документации к этому инструменту.
  2. При восстановлении из файловой системы: Остановите сервер MySQL, замените файлы данных на резервные копии и запустите сервер снова

Практическая работа №20

1. Авторизация — это процесс проверки прав доступа пользователя к определенным ресурсам или действиям в системе. После того как пользователь прошел аутентификацию (подтверждение своей личности, например, через ввод логина и пароля), авторизация определяет, какие действия он может выполнять и к каким данным имеет доступ. Например, администратор может иметь полный доступ к системе, в то время как обычный пользователь может иметь ограниченные права.
2. Регистрация — это процесс создания новой учетной записи пользователя в системе. В ходе регистрации пользователь предоставляет определенную информацию, такую как имя, адрес электронной почты, пароль и другие данные. После успешной регистрации пользователь получает возможность войти в систему и использовать ее функции. Регистрация часто включает в себя подтверждение адреса электронной почты или номера телефона, чтобы удостовериться в подлинности предоставленной информации.  
   1. Управление данными: Хранение информации о пользователях в отдельной таблице позволяет легче управлять данными, такими как обновление профиля, изменение пароля и удаление учетных записей.
   2. Безопасность: Хранение данных о пользователях отдельно от системных пользователей базы данных (например, пользователей MySQL) повышает безопасность. Это позволяет ограничить доступ к самой базе данных и минимизировать риск компрометации.

* 1. Гибкость: Таблица с пользователями может включать дополнительные поля, такие как роль пользователя, дата регистрации, статус аккаунта и другие атрибуты, которые могут быть полезны для управления доступом и функциональностью сайта.
  2. Легкость интеграции: Если вы планируете использовать сторонние сервисы (например, для аутентификации через социальные сети), наличие отдельной таблицы упростит интеграцию.
  3. Масштабируемость: При увеличении числа пользователей отдельная таблица обеспечит лучшую производительность и масштабируемость, чем использование пользователей базы данных.

Практическая работа №21

1. MySqlConnection — это класс, который используется для установления соединения с базой данных MySQL. Он управляет настройками соединения, такими как строка подключения, и отвечает за открытие и закрытие соединения. Этот элемент необходим для выполнения операций с базой данных, таких как выполнение запросов и управление транзакциями.
2. MySqlCommand — это класс, который представляет SQL-команду, которую вы хотите выполнить на сервере MySQL. Он используется для выполнения запросов, таких как SELECT, INSERT, UPDATE и DELETE. В этом элементе можно задавать текст команды, параметры и использовать его для выполнения операций с базой данных через объект MySqlConnection.
3. MySqlDataReader — это класс, который предоставляет возможность чтения данных из базы данных MySQL в виде потока. Он позволяет считывать строки данных, возвращаемые запросами SELECT, и обеспечивает эффективный способ работы с большими объемами данных, так как данные загружаются по мере необходимости. MySqlDataReader работает в режиме только для чтения, что делает его быстрым и легковесным.
4. Для подключения к СУБД MySQL в .NET необходимо подключить пространство имен MySql.Data.MySqlClient

Практическая рвбота №22

* 1. ExecuteNonQuery():
     1. Используется для выполнения SQL-команд, которые не возвращают данные, таких как INSERT, UPDATE и DELETE.
     2. Возвращает количество строк, затронутых выполненной командой.
  2. ExecuteReader():
     1. Используется для выполнения SQL-команд, которые возвращают набор данных, таких как SELECT.
     2. Возвращает объект MySqlDataReader, который позволяет читать строки данных из результата запроса.
  3. ExecuteScalar():
     1. Используется для выполнения SQL-команд, которые возвращают одно значение, например, при использовании агрегатных функций (SUM, COUNT и т.д.).
     2. Возвращает первое значение в первой строке результата запроса.
  4. Создать объект MySqlConnection и открыть соединение.
  5. Создать объект MySqlTransaction с помощью метода BeginTransaction() объекта MySqlConnection.
  6. Привязать MySqlCommand к этой транзакции.
  7. Выполнить необходимые команды.
  8. Подтвердить транзакцию с помощью метода Commit() или откатить с помощью Rollback() в случае ошибки.

1. В Python, для выполнения запросов на модификацию данных в базе данных (например, с использованием библиотеки mysql-connector-python или PyMySQL), обычно используется метод execute() объекта курсора.