МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение   
высшего образования

«КРЫМСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ им. В. И. ВЕРНАДСКОГО»

ФИЗИКО-ТЕХНИЧЕСКИЙ ИНСТИТУТ

Кафедра компьютерной инженерии и моделирования

Отчет

по дисциплине «Базы данных»

студента 3 курса группы ПИ-б-о-221

Бацюра Ильи Алексеевича

направления подготовки 09.03.04 «Программная инженерия»

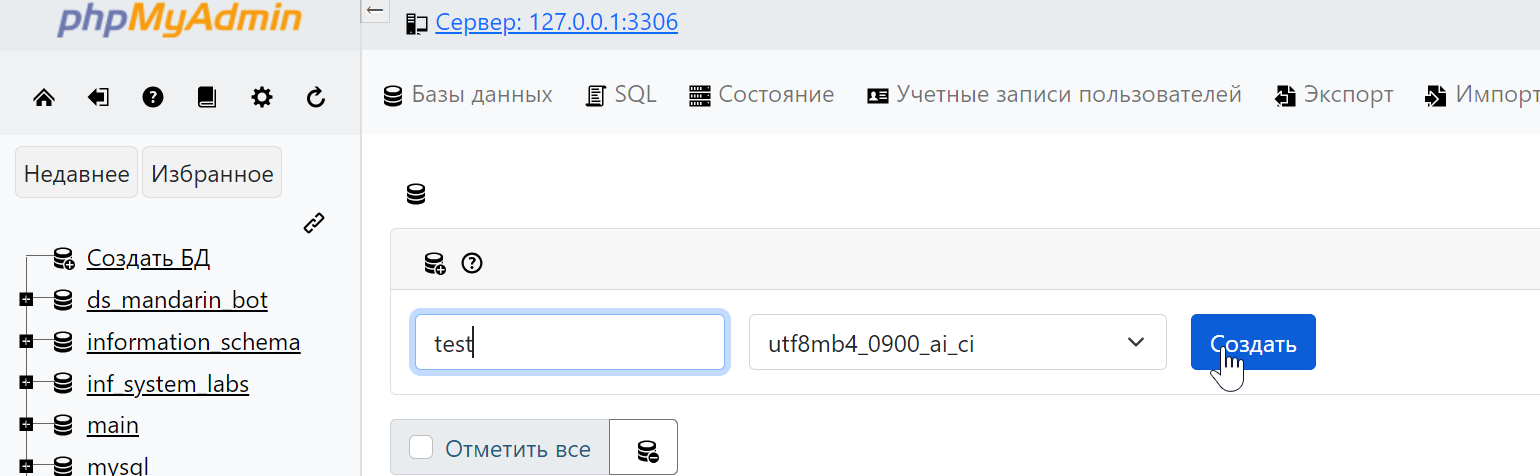
|  |  |  |
| --- | --- | --- |
|  |  |  |

Симферополь, 2024

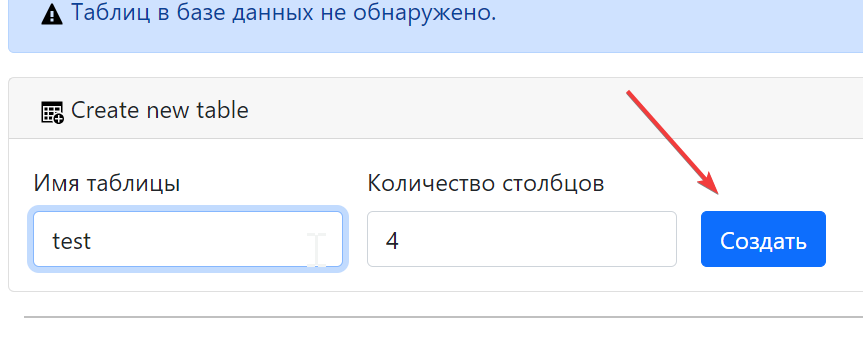
**Лабораторная работа №1**

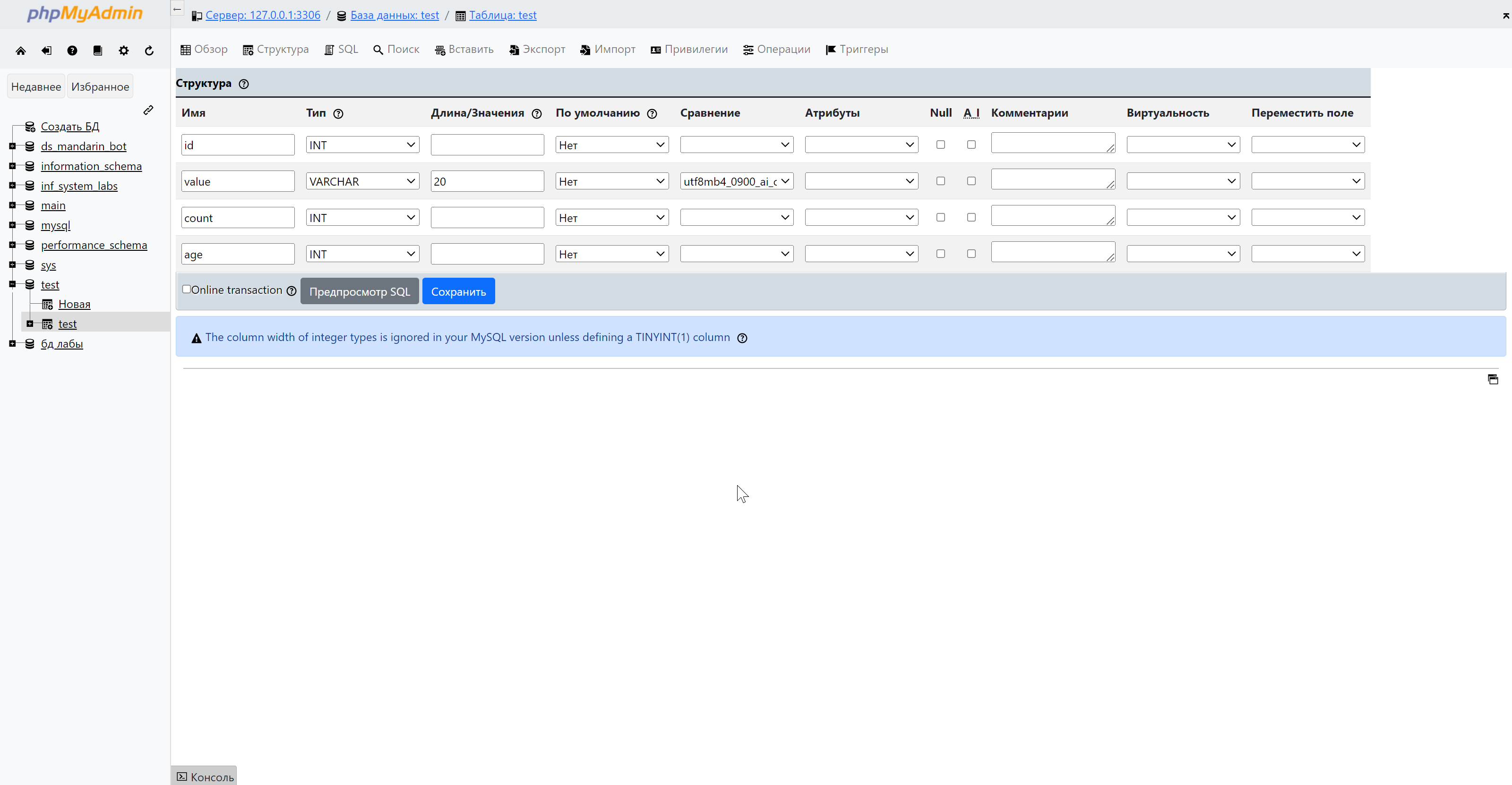
**После инсталяции программы Open Server Panel, можно зайти в СУБД phpMyAdmin и протестировать работоспособность:**

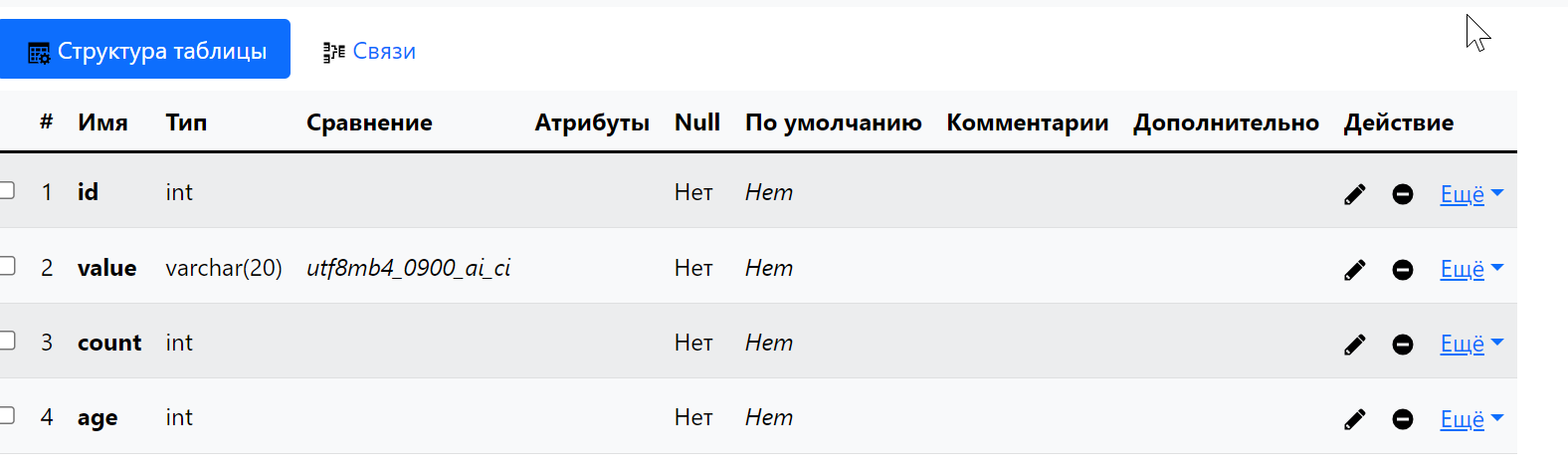
**Создание тестовой базы данных:**



**Создание тестовой таблицы:**







Как видно, программа полностью работоспособна.

**Ответы на контрольные вопросы:**

1. На чем был основан ваш выбор конкретной СУБД.

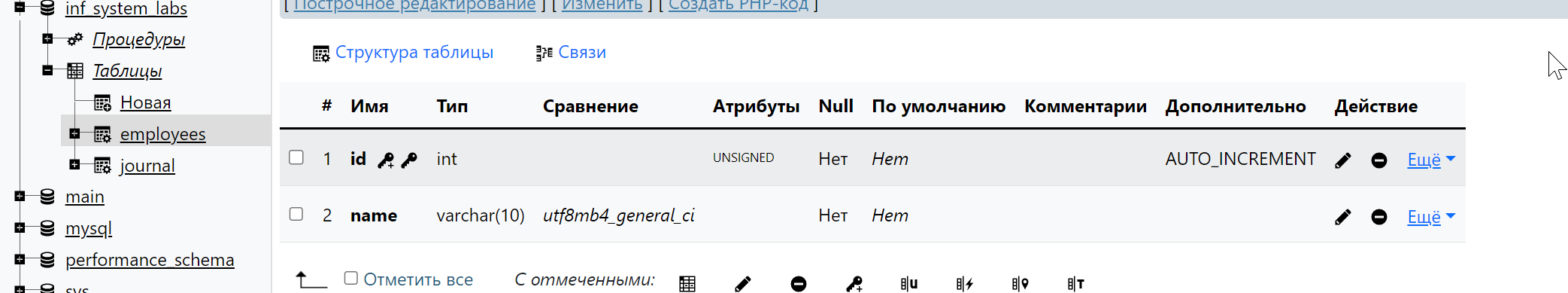
- Мой выбор был основан на том, что по сей день phpMyAdmin остается одной из самых распространённых СУБД, к тому же она довольно удобна на мой взгляд.

2. С какими сложностями вы столкнулись при установке?

- Единственной сложностью было найти подходящую версию Open Server Panel, а именно 5.4.3 с предустановленным модулем phpMyAdmin, т.к. в современной версии данный модуль необходимо устанавливать вручную.

**Лабораторная работа №2**

1. **Создаем таблицу employees, за изменениями которой будем следить.**



1. **Создаем таблицу аудита, в которой мы будем указывать, какие изменения происходят в таблицу employees.**



1. **Создаем процедуру, с помощью которой мы будем добавлять новую запись в таблицу journal.**

DELIMITER \*

CREATE PROCEDURE insertNewJournal(IN \_action VARCHAR(10), IN oldValue VARCHAR(50), IN newValue VARCHAR(50))

BEGIN

INSERT INTO `journal` (`datetime`, `action`, `oldValue`, `newValue`)

VALUES (NOW(), \_action, oldValue, newValue);

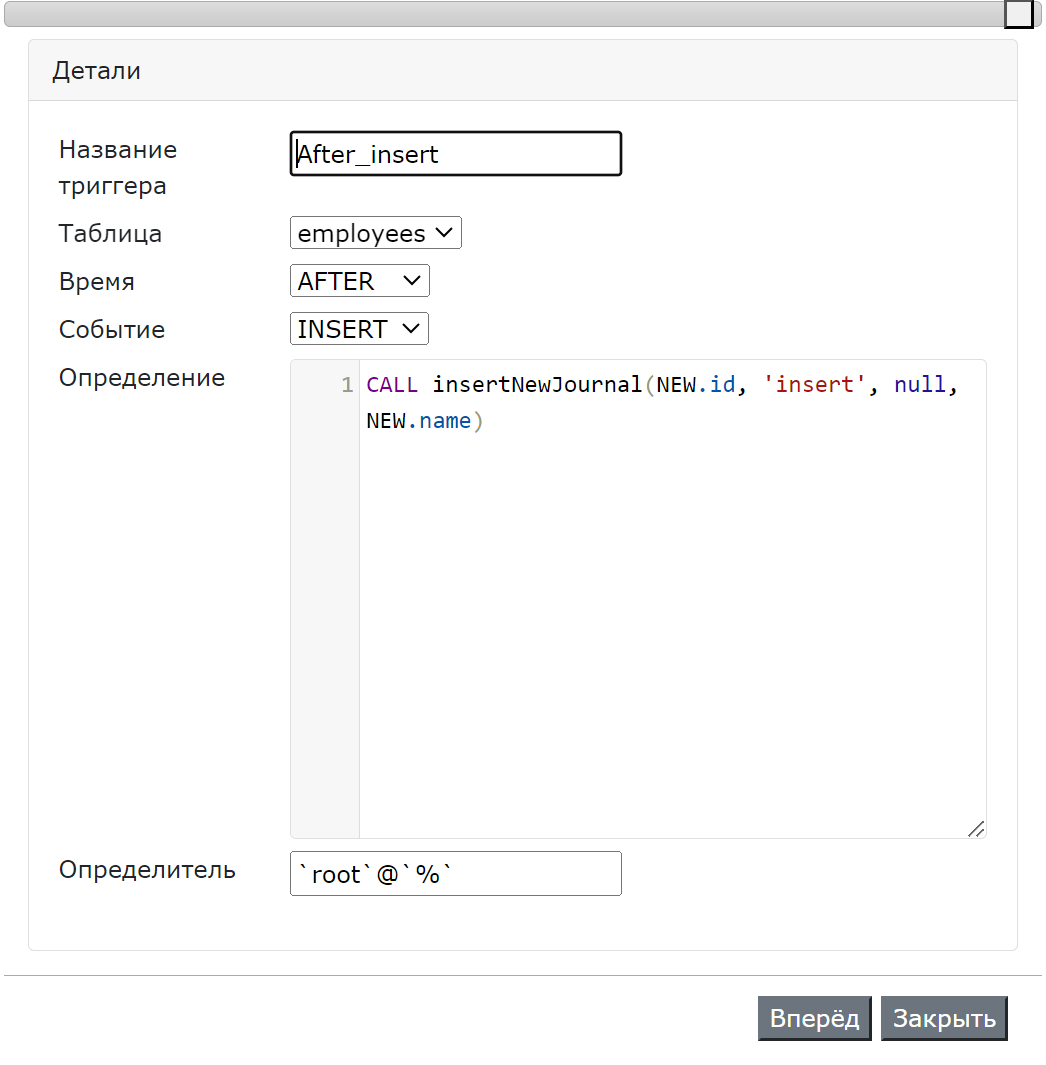
END\*

DELIMITER ;

Здесь, DELIMITER – команда, которая изменяет символ, используемый для разделения SQL-выражений. Мы изменяем символ по умолчанию, чтобы использовать его в теле процедуры и mysql не выдал ошибку. insertNewJournal – название процедуры; \_action, oldValue, newValue – входные параметры процедуры. BEGIN, END – ключевые слова, которые обозначают рамки тела процедуры. В теле процедуры мы создаем новую запись для таблицы journal с текущим временем изменения, самим типом изменения (insert, update, delete), старым значением и новым (если таковые есть). В конце меняем разделяющий знак на знак по умолчанию.

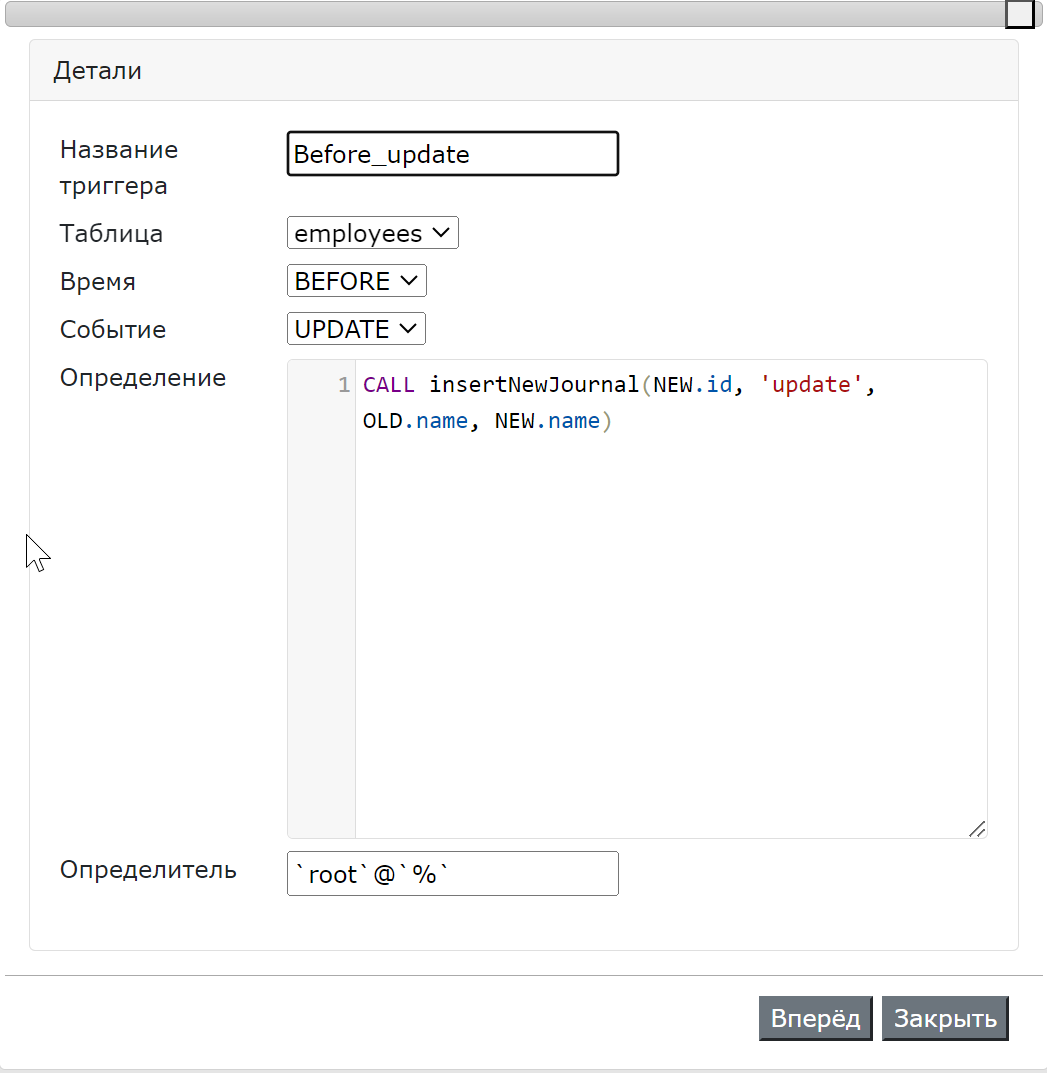
1. **Создаем сами триггеры для таблицы employees.**

**4.1) После вставки**



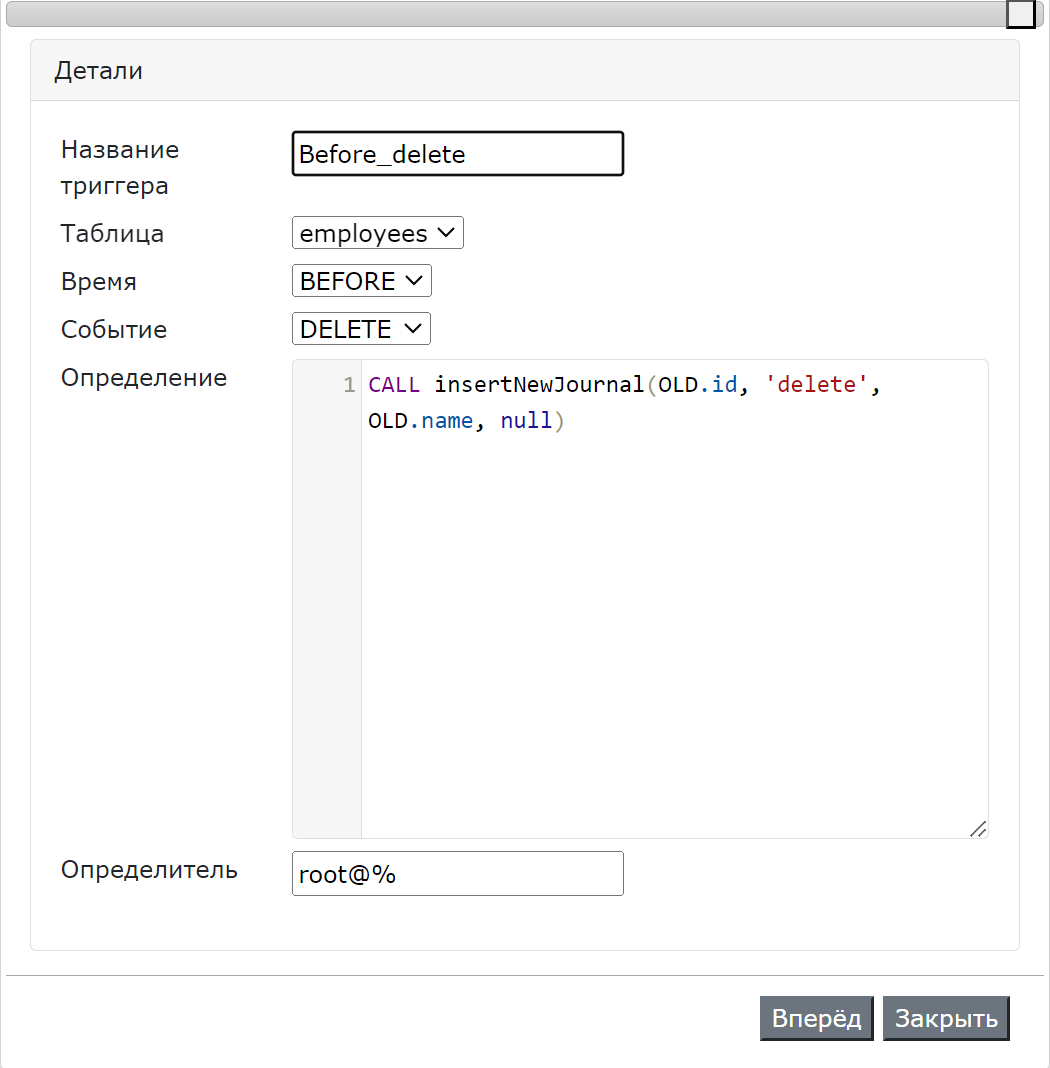
В данном триггере мы вызываем ранее написанную нами процедуру и передаем в нее тип id записи, событие (\_action аргумет) “insert”, в качестве старого значения null (т.к. запись новая, у нее оно отсутствует) и поле с именем работника. (Тип события AFTER Insert я выбрал для того, чтобы в поле id передавался актуальный id, т.к. данное поле является автоинкрементным).

**4.2) Перед изменением**



Здесь мы проделываем то же самое, что и для прошлого триггера, однако в поле со старым значением мы передаем OLD.name и тип события указываем, как “update”.

**4.3) Перед удалением**



В данном триггере мы передаем тип события “delete”, и передаем старые значения id, OLD.name и null в качестве нового имени сотрудника, т.к. в этом событии невозможно новое значение, в т.ч. ключевое слово NEW недоступно.

**Ответы на контрольные вопросы.**

1. Приведите 2-3 примера задач, для решения которых можно использовать триггеры.

- Примеры задач, для решения которых можно использовать триггеры:

* При изменении статуса заказа в таблице "Orders" автоматически обновлять соответствующее поле в таблице "Customers" (например, обновлять поле "LastOrderStatus").
* При добавлении новой записи в таблицу "Employees" автоматически создавать соответствующую запись в таблице "UserAccounts" с уникальным идентификатором и начальным паролем.

2. Можно ли привязать один триггер к двум разным таблицам?

- Нет, данная операция не возможна. В MySQL нельзя привязать один триггер к двум разным таблицам. Каждый триггер привязан к определенной таблице и срабатывает в ответ на события, происходящие с этой таблицей. Если необходимо выполнить одну и ту же операцию при изменении данных в двух разных таблицах, то следует создать два отдельных триггера, каждый из которых будет срабатывать на соответствующей таблице.

**Лабораторная работа №3**

**Создание первой процедуры:**

DELIMITER $$

CREATE PROCEDURE `addStudent`(IN name VARCHAR(10))

BEGIN

DECLARE max\_id INT;

INSERT INTO `students` (`name`) VALUES (name);

SET max\_id = (SELECT MAX(`id`) FROM `students`);

UPDATE `students` SET `mark`=FLOOR((RAND() \* 100) % 101) WHERE id=max\_id;

END; $$

DELIMITER ;

Данная процедура принимает в качестве аргумента 1 параметр - name (имя студента). В теле процедуры мы объявляем переменную max\_id – отвечает за хранение последнего id в таблице. Далее мы добавляем новую запись в таблицу с студентов с именем name и с пустыми баллами. После этого мы при помощи функции MAX ищем макс. id (то есть последнего добавленного нами студента) и сохраняем в переменную max\_id. Наконец, мы назначаем студенту с max\_id случайно сгенерированные баллы при помощи функции RAND. Таким образом, мы создали процедуру для добавления студента со случайными баллами (диапазон от 0-100 включительно).

**Создание второй процедуры:**

DELIMITER $$

CREATE PROCEDURE `average2Mark`(IN mark1 INT, IN mark2 INT)

BEGIN

SELECT (mark1 + mark2) / 2 AS `Средний балл`;

END; $$

DELIMITER ;

Данная процедура принимает 2 числовых аргумента: mark1 и mark2 (баллы студентов), рассчитывает из них средний балл и выводит результат пользователю при помощи ключевого слова SELECT. Также при помощи AS псевдонима мы переименовываем колонку результата в `Средний балл`.

**Ответы на контрольные вопросы:**

1. Можно ли использовать в хранимой процедуре два SQL запроса типа UPDATE и SELECT? Если да, то каким будет результат выполнения такой процедуры?

- Да, в хранимой процедуре можно использовать несколько SQL запросов. В данном случае, можно использовать как UPDATE, так и SELECT запросы.

Результат выполнения такой процедуры будет зависеть от последовательности и логики запросов.

1. Если сначала выполняется UPDATE запрос, то он изменит данные в базе.
2. Затем, если выполняется SELECT запрос, то он вернёт результат на основе уже изменённых данных.

Таким образом, результат SELECT запроса будет отражать состояние данных после выполнения UPDATE запроса.

Однако, стоит отметить, что в некоторых СУБД (например, в MySQL) изменения, внесённые UPDATE запросом, могут не быть видны в SELECT запросе, выполняемом в той же транзакции, из-за механизма "repeatable read isolation level". В таком случае, чтобы увидеть изменения, необходимо зафиксировать транзакцию (COMMIT) после UPDATE запроса.

2. Можно ли использовать в хранимой процедуре два SQL запроса типа SELECT? Если да, то каким будет результат выполнения такой процедуры?

- Да, в хранимой процедуре можно использовать несколько SQL запросов типа SELECT. Результат будет зависеть от логики и последовательности запросов, и может быть представлен в виде нескольких наборов данных. Если в процедуре используются два SELECT запроса, то результат выполнения процедуры может быть представлен в виде двух наборов данных, которые могут быть обработаны или отображены отдельно.

3. Какие типы данных можно использовать для аргументов хранимой процедуры?

- Типы данных, которые можно использовать для аргументов хранимой процедуры, зависят от конкретной СУБД (системы управления базами данных), которую вы используете.

Общие типы данных, которые могут быть поддерживаемыми большинством СУБД, включают:

1. Целочисленные типы данных: INT, BIGINT, SMALLINT, TINYINT и т.д.
2. Числа с плавающей точкой: FLOAT, DOUBLE, DECIMAL и т.д.
3. Строковые типы данных: CHAR, VARCHAR, TEXT, NCHAR, NVARCHAR, NTEXT и т.д.
4. Дата и время: DATE, TIME, DATETIME, TIMESTAMP и т.д.
5. Логический тип данных: BOOLEAN, BIT и т.д.
6. Другие типы данных: BINARY, VARBINARY, ARRAY, JSON, XML и т.д.

Кроме того, некоторые СУБД также поддерживают пользовательские типы данных, такие как перечисления (enum) или структуры (struct).

**Лабораторная работа №4**

DELIMITER $$

CREATE PROCEDURE `increasePrice`(IN brand VARCHAR(50), IN count INT UNSIGNED)

BEGIN

DECLARE done INT DEFAULT 0;

DECLARE current\_brand VARCHAR(50);

DECLARE current\_price INT UNSIGNED;

DECLARE carscur CURSOR FOR

SELECT `brand`, `price` FROM `cars`;

DECLARE CONTINUE HANDLER FOR SQLSTATE '02000' SET done=1;

OPEN carscur;

FETCH carscur INTO current\_brand, current\_price;

WHILE done = 0 DO

IF current\_brand = brand THEN

UPDATE cars SET `price` = current\_price + count WHERE cars.`brand` = brand;

END IF;

FETCH carscur INTO current\_brand, current\_price;

END WHILE;

CLOSE carscur;

END; $$

DELIMITER ;

Данная процедура принимает 2 аргумента: brand – строковый тип, фирма производитель машины; count – числовой тип, сумма, на которую подорожала машина данной фирмы. В начале мы объявляем переменные для курсора, далее в цикле проходимся по каждой записи и если бренд текущей записи соответствует бренду, который мы указали при вызове процедуры, то цена данной записи увеличится на значение, которое мы также указали при вызове. В результате все машины конкретной фирмы будут дороже на значение count.

**Ответы на контрольные вопросы:**

1. В каких случаях в хранимых процедурах следует применять язык хранимых процедур вместо SQL-запросов?

- В некоторых случаях, таких как выполнение сложных операций, управление транзакциями, повышение производительности и обеспечение безопасности, использование языка хранимых процедур может быть более удобным и эффективным, чем SQL-запросы. Это связано с тем, что язык хранимых процедур позволяет более точно управлять потоком выполнения операций, обрабатывать исключения, управлять транзакциями и ограничивать доступ к данным.

2. Можно ли считать язык хранимых процедур Тьюринг-полным?

- Нет, язык хранимых процедур не является Тьюринг-полным.

Тьюринг-полнота означает, что язык программирования может быть использован для выполнения любой вычислимой задачи. Язык хранимых процедур имеет ограниченные возможности по сравнению с языками программирования общего назначения, такими как Python, Java или C++.

Например, язык хранимых процедур не поддерживает рекурсию, что означает, что невозможно определить функцию, которая вызывает саму себя. Кроме того, язык хранимых процедур имеет ограниченные возможности по работе с памятью, что делает невозможным реализацию некоторых алгоритмов, требующих динамического выделения памяти.

Таким образом, язык хранимых процедур не является Тьюринг-полным и не может быть использован для решения всех возможных вычислительных задач. Он предназначен для выполнения определенных задач в рамках работы с базами данных и имеет определенные ограничения по сравнению с языками программирования общего назначения.