|  |  |
| --- | --- |
|  | |
| МИНОБРНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** | |
| Институт информационных технологий (ИТ) |
| Кафедра корпоративных информационных систем |

|  |  |
| --- | --- |
| **Отчет по практическим работам** | |
| по дисциплине «Информационное обеспечение предприятия» | |
|  | |
|  | |
| Выполнила студентка группы ИКБО-08-20 | Подвойская А. В. |
| Принял | Крюков Д. А. |

|  |  |  |
| --- | --- | --- |
| Практические работы выполнены | «\_\_»\_\_\_\_\_\_\_2023 г. |  |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_2023 г. |  |

2023 г.

**Оглавление**

[Практическая работа №1 3](#_gjdgxs)

[Практическая работа №2 5](#_30j0zll)

[Практическая работа №3 9](#_1fob9te)

[Практическая работа №4 14](#_3znysh7)

[Практическая работа №5 17](#_2et92p0)

[Практическая работа №6 21](#_tyjcwt)

[Практическая работа №7 27](#_3dy6vkm)

[Практическая работа №8 31](#_1t3h5sf)

[Практическая работа №9 34](#_4d34og8)

[Практическая работа №10 37](#_2s8eyo1)

[Практическая работа №11 41](#_17dp8vu)

[Практическая работа №12 44](#_3rdcrjn)

[Практическая работа №13 48](#_26in1rg)

[Практическая работа №14 54](#_lnxbz9)

[Практическая работа №15 58](#_35nkun2)

[Практическая работа №16 59](#_1ksv4uv)

[Практическая работа №17 64](#_44sinio)

# Практическая работа №1

Выбранная поисковая система: Chrome. Документация: [https://itandlife.ru/technology/sekrety-poiska-v-google ili-yazyk-poiskovyx-zaprosov-google/](https://itandlife.ru/technology/sekrety-poiska-v-google%20ili-yazyk-poiskovyx-zaprosov-google/)

Запросы, содержащие операторы «И», «ИЛИ», «НЕ», скобки:

1. "Лучшие курорты И (море ИЛИ горы) НЕ Таиланд" – данный запрос позволяет найти лучшие курорты у моря или в горах, исключая при этом Таиланд.
2. "(Фильмы ИЛИ сериалы) И (2021 ИЛИ 2022) НЕ ужасы – данный запрос позволяет найти информацию фильмах или сериалах, вышедших в 2021 или 2022 году, исключая фильмы из жанра ужасов.
3. "Здоровая диета И (веганская ИЛИ палео ИЛИ сыроедческая) НЕ глютен" – данный запрос позволяет найти информацию о здоровых диетах, таких как веганская, палео или сыроедческая, исключая продукты, содержащие глютен.
4. "Новости И (политика ИЛИ экономика)" – данный запрос позволяет найти новости, связанные с политикой или экономикой.
5. "((Лучшие практики И советы) ИЛИ (Эффективные стратегии)) (продаж И Маркетинга)" – данный запрос позволяет получить информацию о лучших практиках, советах или эффективных стратегиях в области продаж и маркетинга.

Критерии, которым должны удовлетворять системы семантического поиска:

1. **Текущие тенденции.** После выбора президента на запрос: «Кто стал новым президентом?», поиск должен понять запрос и дать соответствующие результаты, основанные на текущих тенденциях и новостях.
2. **Местонахождение пользователя.** Если человек ввел запрос «Какая сейчас температура?», семантическая поисковая система должна предоставить результаты, основанные на месте его нахождения в момент запроса.
3. **Цель поиска.** Семантический поиск основывается на выдаче подходящих результатов, основанных на цели поиска пользователя, а не определенных слов, использованных при обращении к поисковой системе.
4. **Вариации слов в семантическом поиске.** Поиск должен учитывать лингвистические особенности запроса. Например видеть разницу, в таких словах как «техника» (транспортные средства или должность в род. падеже), «техники» (методики или должность во мн.ч.), «техник» (должность или методики во мн.ч).
5. **Синонимы**. Система должна понимать синонимы и давать похожие результаты на любые синонимичные слова, запрашиваемые пользователем.
6. **Простой язык при запросах**. Система должна понимать даже технически неправильно составленные запросы.
7. **Зависимость между значением фразы и использованными в ней словами**. Конкретные слова в словосочетаниях и фразах или их порядок могут изменить истинное значение всего запроса.

Сравнительная таблица для поисковых систем Яндекс и Google.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Запрос | Яндекс | | Google | |
| Достоверность | Полнота | Достоверность | Полнота |
| Квадрат скорости света | 1 | 50% | 1 | 50% |
| Гипотенуза треугольника с катетами 3 и 4 | 1 | 50% | 1 | 35% |
| Бетельгейзе ярче солнца? | 0 | 0% | 1 | 50% |
| 5х-10 = 0, чему равно х? | 1 | 50% | 1 | 40% |
| Может ли хорда быть больше диаметра? | 1 | 50% | 1 | 45% |
|  | 0,8 | 40% | 1 | 44% |

# Практическая работа №2

# Практическая работа №3

Функционал штатного диагностического инструмента Chrome:

1. Инспектор элементов (**Elements**): позволяет анализировать и изменять HTML и CSS элементы в режиме реального времени.

– Styles: показывает и позволяет менять CSS-правила, видеть вычисляемые значения CSS-свойств для выбранного элемента и унаследованные значения. Обеспечивает быстрый переход к источнику стилей, предоставляет инструменты для визуализации границ элементов, отступов и размеров, позволяет добавлять и удалять классы.

– Computed: позволяет просматривать наследуемые CSS-свойства и видеть их актуальные значения для выбранного элемента, показывает иерархическую информацию о примененных стилях и унаследованных значениях для каждого свойства CSS, а так же вычисляемые стили по умолчанию, предоставляет возможность сортировки и фильтрации вычисляемых стилей.

– Layout: позволяет просматривать и менять размеры, позиции и геометрию элементов на странице, включая их ширину, высоту, отступы, положение и другие свойства. При включенном режиме отображения CSS Box Model можно увидеть границы элементов относительно других элементов на странице. Позволяет просматривать плоский список элементов на странице и анализировать их расположение.

– Event Listeners: отображает список событий и код обработчиков событий, которые привязаны к выбранному элементу на странице, позволяет менять, удалять, создавать и фильтровать события.

– DOM Breakpoints: позволяет добавлять и удалять точки останова, а также управлять ими. Break on Subtree Modifications позволяет установить точку останова на изменениях внутри определенного DOM-элемента или его поддерева. Break on Attribute Modifications устанавливает точку останова на изменение атрибута конкретного DOM-элемента. Break on Node Removal позволяет установить точку останова при удалении указанного DOM-элемента или его дочерних элементов. Break on Attribute Removal установливает точку останова на удаление определенного атрибута с DOM-элемента.

– Properties: позволяет просматривать и менять список всех доступных свойств для выбранного элемента, а также отображает наследуемые и вычисляемые значения свойств стилей. Предоставляет возможность фильтрации свойств.

– Accessibility: сканирует веб-страницу и выявляет возможные проблемы с доступностью, такие как отсутствие атрибутов доступности, используемые цветовые комбинации, отсутствие альтернативного текста для изображений и другие. Позволяет просматривать атрибуты доступности, примененные к выбранным элементам на странице, что включает атрибуты с описанием контента, использование меток для связи элементов, атрибуты языка и другие. Предоставляет полезные советы и рекомендации по улучшению доступности веб-страницы. Можно эмулировать ограничения пользователей с определенными нарушениями зрения или движения, чтобы оценить, как веб-страница будет восприниматься и использоваться такими пользователями.

2. Консоль (**Console**): является окном для выполнения JavaScript-кода и вывода ошибок или сообщений. Позволяет изменять и взаимодействовать с веб-страницей в режиме реального времени. Можно изменять значения CSS-свойств, модифицировать DOM-элементы, применять новые стили или захватывать события. Позволяет фильтровать и искать выводимые сообщения. Позволяет профилировать JavaScript-код, т.е. анализировать его производительность и время выполнения, можно найти узкие места в коде и оптимизировать его для более эффективной работы.

3. Источники (**Sources**): позволяет просматривать и редактировать исходный код веб-страницы, включая файлы JavaScript, CSS и HTML. Предоставляет возможности отладки и фильтрации. Позволяет просматривать и редактировать файлы CSS, применяемые к веб-странице. Позволяет работать с исходными картами (source maps), которые связывают минифицированный или скомпилированный код с исходным кодом. Внесенные изменения в исходный код можно сохранить локально или применить непосредственно к ресурсу на сервере.

4. Сеть (**Network**): позволяет отслеживать все сетевые запросы, отправленные веб-страницей (HTTP, HTTPS). Можно видеть информацию о каждом запросе, такую как URL, метод, код ответа, время загрузки, размер ответа и другие свойства. Предоставляет информацию о времени загрузки каждого запроса и его компонентов. Дает возможность фильтровать сетевые запросы по различным параметрам, таким как тип запроса, код ответа, домен и другие. Предоставляет возможность просмотреть подробности каждого запроса, включая заголовки, тело запроса, параметры, cookies и другую информацию. Позволяет эмулировать различные сценарии сети, чтобы проверить работу веб-страницы при различных условиях (например, медленное соединение или потери пакетов). Предусмотрена возможность сохранения сетевых запросов в виде файлов HAR (HTTP Archive) для последующего анализа или сравнения.

5. (**Performance**): позволяет измерять время, затраченное на разные этапы загрузки и выполнения страницы. Можно видеть, сколько времени занимает загрузка ресурсов, разбиение по этапам, время выполнения JavaScript и другие параметры, чтобы идентифицировать узкие места и оптимизировать производительность. Отображает графическую визуализацию процесса загрузки и выполнения страницы. Позволяет профилировать производительность кода JavaScript. Можно анализировать использование CPU, трассировать функции, проверять объем памяти и т.д.

6. Память (**Memory**): позволяет записывать профиль памяти веб-приложения во время его работы для анализа утечки памяти, определить использование ресурсов и выяснить, какие объекты занимают больше всего памяти. отображает информацию о текущем использовании памяти веб-приложением. Можно видеть общий объем памяти, используемый различными типами объектов, такими как DOM-узлы, JavaScript-объекты, изображения и другие. Позволяет профилировать процесс выделения и освобождения памяти, анализируя время жизни объектов. Позволяет генерировать отчеты о использовании памяти и утечках.

7. Приложение (**Application**): позволяет просматривать и взаимодействовать с различными хранилищами данных на стороне клиента, такими как Local Storage, Session Storage и IndexedDB. Можно просматривать существующие данные, добавлять, обновлять или удалять данные, а также анализировать использование памяти. позволяет управлять и анализировать кэширование ресурсов (Cache Storage) для веб-приложения. Можно просматривать существующие кэшированные ресурсы, добавлять или удалять их, а также выполнять операции обновления кэша. Предоставляет просмотр и взаимодействие с Web Storage API (Local Storage и Session Storage). Позволяет просматривать состояние Service Workers, перезагружать или отключать их. Предоставляет возможность просматривать и анализировать файл манифеста приложения (web app manifest) для прогрессивных веб-приложений (PWA). Он содержит информацию о приложении, такую как название, значок, цветовую схему и другие метаданные. Дает возможность работать с файловой системой веб-приложения, поэтому можно просматривать файлы и папки, создавать, изменять или удалять их.

8. Безопасность (**Security**): позволяет просматривать данные о сертификатах SSL/TLS, используемых на веб-странице. Можно проверить, срок действия сертификата, его валидность и цепочку доверия. Проводит проверку на наличие уязвимостей безопасности на веб-странице и предупреждает о потенциальных проблемах, таких как смешанные ресурсы (mixed content), неиспользуемые сертификаты, уязвимости кросс-сайтового скриптинга, захват сеансов и другие. Проверяет безопасность сетевых связей на веб-странице, таких как HTTP, HTTPS и других протоколов. Обеспечивает безопасность кросс-сайтовых запросов. Предупреждает о возможных уязвимостях межсайтового скриптинга и предоставляет инструменты для их обнаружения и исправления. Предоставляет возможность просматривать информацию о правах доступа, запрошенных веб-страницей, таких как изображения, местоположение, камера и другие.

9. Инструмент аудита (**Lighthouse**): проводит аудит производительности веб-страницы, оценивая факторы, такие как время загрузки, время первой визуальной отрисовки, скорость загрузки страницы, компрессия ресурсов, использование кэша и другие метрики. Проводит аудит доступности веб-страницы, проверяя соответствие стандартам доступности WCAG и обнаруживая потенциальные проблемы, такие как отсутствие альтернативного текста для изображений, недостаточный контраст цветов, отсутствие меток для связывания элементов и другие. Оценивает совместимость веб-страницы с мобильными устройствами, давая оценку мобильной дружественности, оптимизацию мобильного просмотра, скорость загрузки на мобильном устройстве и другие аспекты. Проводит аудит прогрессивных веб-приложений, оценивая степень соответствия стандартам PWA, таких как манифест веб-приложения, наличие сервисных работников. Генерирует подробные отчеты о выполненном аудите с оценками и рекомендациями для каждого аспекта.

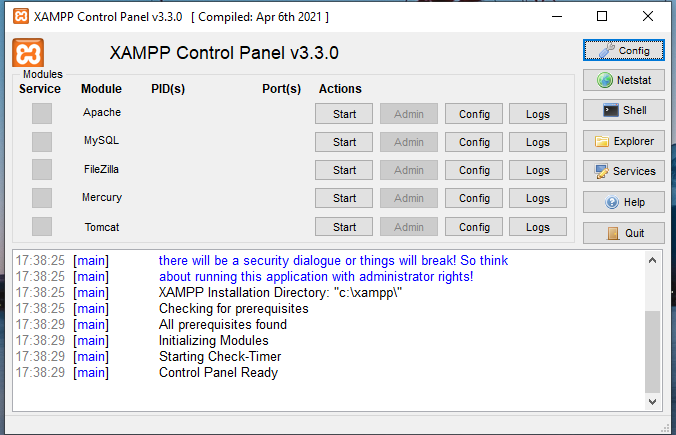
Таким образом, панель разработчика является крайне полезной в web-разработке, т.к. предоставляет огромный функционал. Позволяет анализировать, редактировать и отлаживать HTML, CSS и JavaScript файлы на лету, предоставляя возможности фильтрации, что оптимизирует работу. Анализирует безопасность, скорость и производительность кода. Позволяет имитировать ситуации, такие как пользователь с ограниченными возможностями или же потеря пактов. Составляет отчеты по аудиту и использованию памяти. Генерирует рекомендации по улучшению доступности веб-страницы.

# Практическая работа №4

1. Получить инсталлятор XAMPP с официального сайта;

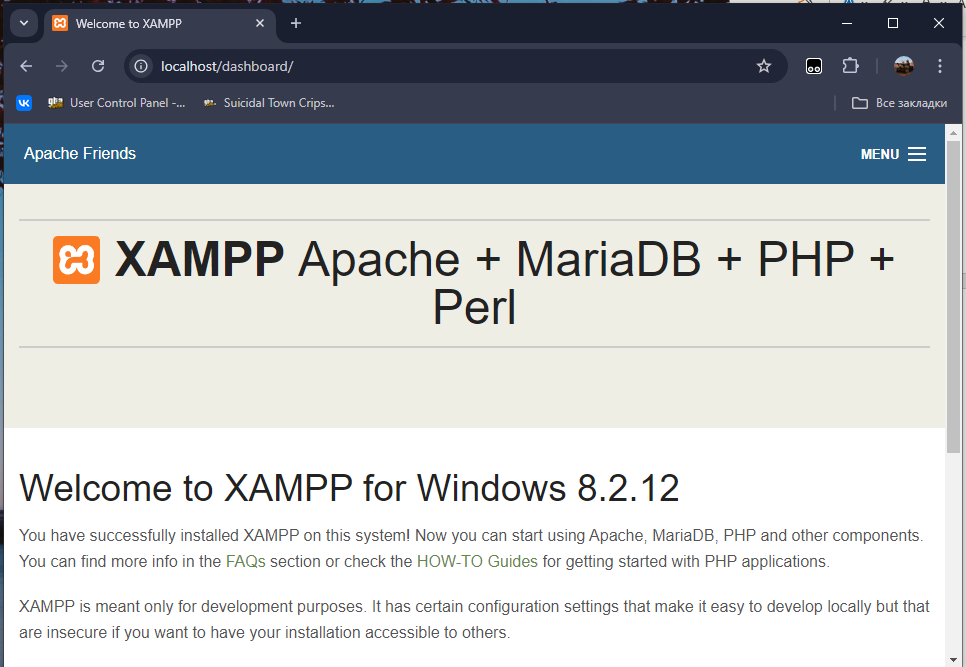
2. Установить сборку по пути «C:/xampp» компонентами: Apache, MySQL, PHP, phpMyAdmin;

3. После установки запустить компоненты Apache и MySQL;

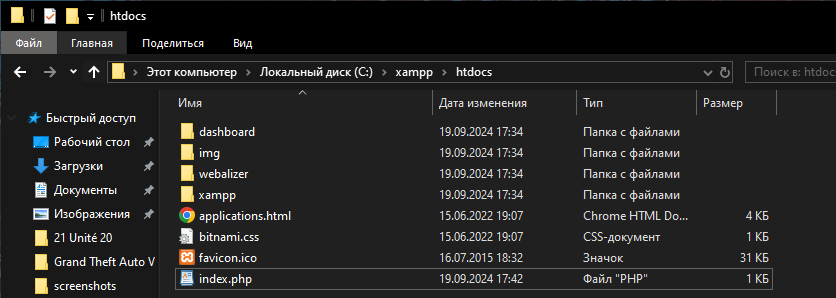


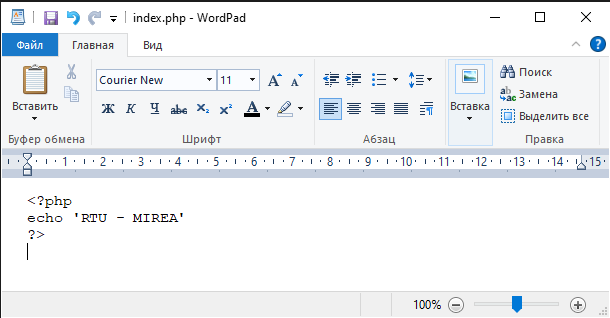
4. Проверить работу web-сервера Apache: открыть браузер и проверить адрес «http://localhost/». Получить уведомление об успешной установке и запуске;

Обращение к localhost

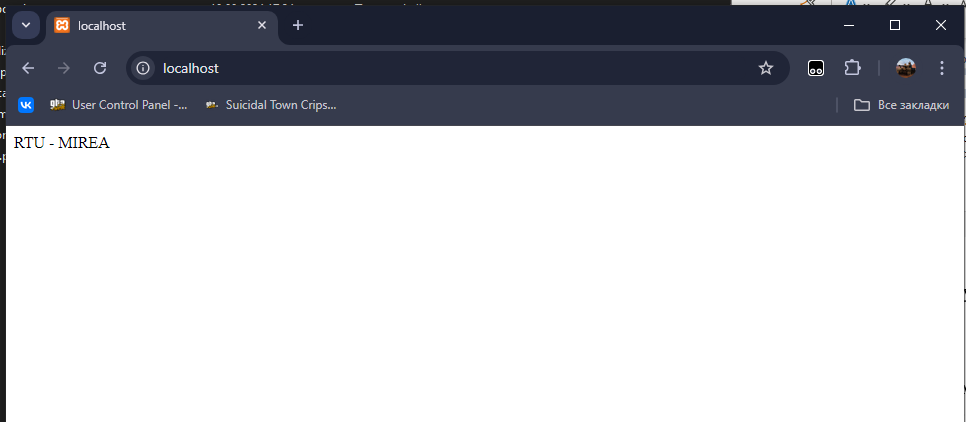


5. Определить корневую директорию тестового web-сайта. В корневой директории создать простой тестовый сайт и начальный файл webсайта index.php, содержащий вывод сообщения;



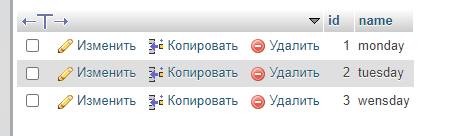


6. Проверить работоспособность web-сайта, перейдя в браузере по его адресу;

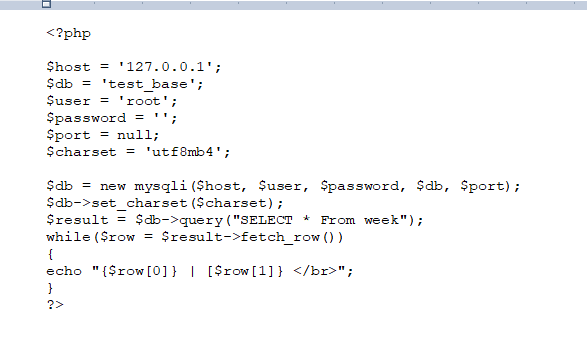
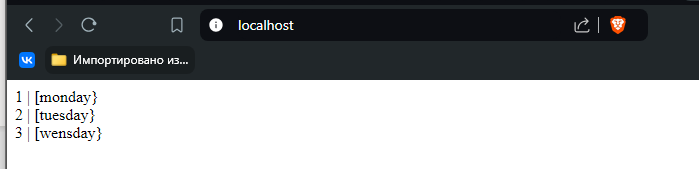


7. С использованием phpMyAdmin создать новую базу данных, таблицу с двумя столбцами. Первый столбец: тип данных - integer, индекс – primary, auto\_increment. Второй столбец типа text;

8. Наполнить таблицу записями;



9. Добавить в код файла index.php скрипт, содержащий инструкции подключения к СУБД, выполнения запроса select …, сохранения результата запроса в массив, вывода записей на web-страницу, отключения от СУБД.

# Практическая работа №5

1. На базе инфраструктуры, полученной в результате выполнения практического задания №4, определить скриптовый язык для реализации взаимодействия с базой данных: **PHP**, Perl, Pyton, NodeJS, ASP.net и др. Определить СУБД: MySQL, SQLite, PostgreSQL, MariaDB, Firebird;

2. Добавить в интерпретатор скриптового языка соответствующую библиотеку для работы с СУБД, импортировать библиотеку в код – встроен в PHP;

3. Реализовать в скрипте функции подключения к СУБД, закрытия соединения, Создания базы данных, создания таблиц базы данных, наполнение таблиц баз данных, вывод с объединением данных в таблицах;

<?php

class SQL

{

    private $host     = '127.0.0.1:4445';

    private $user     = 'root';

    private $password = '';

    private $database;

    private $db\_name;

    function connect()

    {

        $this->database = new mysqli($this->host, $this->user, $this->password);

        $this->database->autocommit(true);

    }

    function disconnect($drop = true)

    {

        if($this->database)

        {

            if($drop)

            {

                $this->database->query("DROP DATABASE IF EXISTS {$this->db\_name}");

            }

            $this->database->close();

        }

    }

    function create\_database($name)

    {

        $this->db\_name = $name;

        $this->database->query("CREATE DATABASE IF NOT EXISTS {$name}");

        $this->database->query("USE {$name}");

    }

    function create\_table($name, $vars)

    {

        $this->database->query("CREATE TABLE IF NOT EXISTS {$name} ({$vars})");

    }

    function insert\_data($db, $vars)

    {

        for($i = 0; $i < count($vars); ++$i) {

            if(gettype($vars[$i]) == "string")

            {

                $vars[$i] = '"'."{$vars[$i]}".'"';

            }

            elseif(gettype($vars[$i]) == "NULL")

            {

                $vars[$i] = 'NULL';

            }

        }

        $var\_str = implode(', ', $vars);

        $this->database->query("INSERT INTO {$db} VALUES ({$var\_str}) ");

    }

    function data\_query($query)

    {

        echo $query;

        $result = $this->database->query($query);

        $retval = array();

        while($row = $result->fetch\_row())

        {

            $temp = array();

            foreach($row as $element)

            {

                array\_push($temp, strval($element));

            }

            array\_push($retval, $temp);

        }

        return $retval;

    }

    function get\_data($table\_name)

    {

        return $this->data\_query("SELECT \* FROM {$table\_name}");

    }

    function get\_data\_join($table\_name1, $table\_name2, $key1, $key2, $cols = "\*")

    {

        return $this->data\_query("SELECT {$cols}

        FROM {$table\_name1}

        JOIN {$table\_name2} ON {$table\_name1}.{$key1}={$table\_name2}.{$key2}");

    }

    function get\_data\_join\_many\_to\_many($table\_name1, $table\_name2, $table\_name3, $key1, $key2, $col)

    {

        return $this->data\_query("

        SELECT Merged.{$col}, {$table\_name2}.{$col}

        FROM (SELECT {$table\_name1}.{$key1}, {$table\_name3}.{$key2}, {$table\_name1}.{$col}

                FROM {$table\_name1}

                JOIN {$table\_name3} ON {$table\_name1}.{$key1}={$table\_name3}.{$key1}) AS Merged

        JOIN {$table\_name2} ON {$table\_name2}.{$key2}=Merged.{$key2}");

    }

}

function show\_table($table)

{

    echo "<table style='border: 1px solid black;'><tboby>";

    foreach($table as $row) {

        echo "<tr>";

        foreach($row as $val)

        {

            echo "<td style='border: 1px solid black; padding: 10px;'>".strval($val).'<br>';

        }

        echo "</tr>";

    }

    echo "</tboby></table></br>";

}

$db = new SQL();

$db->connect();

$db->create\_database("work");

$db->create\_table("employer", "Id\_employer INT PRIMARY KEY AUTO\_INCREMENT, Name VARCHAR(100)");

$db->insert\_data("employer", [NULL, "Pasha"]);

$db->insert\_data("employer", [NULL, "Katya"]);

$db->create\_table("job", "Id\_job INT PRIMARY KEY AUTO\_INCREMENT, Name VARCHAR(100)");

$db->insert\_data("job", [NULL, "Programmer"]);

$db->insert\_data("job", [NULL, "Tester"]);

$db->insert\_data("job", [NULL, "Artist"]);

$db->create\_table("sellary", "Id\_sellary INT, Price FLOAT");

$db->insert\_data("sellary", [1, 150000]);

$db->insert\_data("sellary", [2, 120000]);

$db->insert\_data("sellary", [3, 130000]);

$db->create\_table("employer\_job", "Id\_employer INT, Id\_job INT");

$db->insert\_data("employer\_job", [2, 1]);

$db->insert\_data("employer\_job", [1, 2]);

show\_table($db->get\_data("employer"));

show\_table($db->get\_data("job"));

show\_table($db->get\_data("sellary"));

show\_table($db->get\_data("employer\_job"));

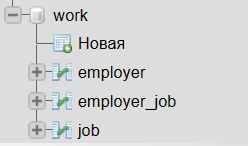
show\_table($db->get\_data\_join("job", "sellary", "Id\_job", "Id\_sellary", "job.Name, sellary.Price"));

show\_table($db->get\_data\_join\_many\_to\_many("employer", "job", "employer\_job", "Id\_employer", "Id\_job", "Name"));

$db->disconnect();

?>

4. Создать две таблицы, объединенные связью «многие ко многим». Например, автомобили-водители, отели-туристы, заказчики-исполнители, фильмы-актеры, писатели-книги и т.п. Создать таблицу-справочник для связи «один ко многим»;



5. Наполнить таблицы баз данных осмысленными значениями;

6. Выполнить с помощью скрипта примеры вывода данных, объединенных по трем таблицам;



7. Составить схему базы данных с использованием MySQL Workbench;

