Работа с базами данных SQL в PHP

Что такое база данных

База данных (сокращенно БД) - это место, в котором хранятся данные сайта.

Это могут быть тексты страниц, списки пользователей с их логинами и паролями, каталоги продукции и другое.

База данных состоит из **таблиц**. Что такое таблица вы знаете из жизни: это **строки и столбцы**. А на пересечении строк и столбцов располагаются **ячейки**.

В базах данных столбцы часто называют **полями**.

Это легко можно вообразить себе, представив документ **Excel**. Базой данных будет являться сам **документ** (книга), а таблицами - каждый **лист** этой книги.

PhpMyAdmin

Для редактирования баз данных часто пользуются программой PhpMyAdmin.

PhpMyAdmin (читается PHPмайадмин, часто пишут аббревиатуру PMA или ПМА) - это оболочка для работы с базами данных прямо у вас в браузере.

Вы можете править содержимое таблиц, создавать новые базы данных и таблицы - и все это через веб-интерфейс, не зная SQL.

Задачи на PhpMyAdmin

Давайте откроем **PhpMyAdmin** и приступим к изучению его интерфейса.

Чтобы вам было проще с ним освоится, сделайте следующие практические задачи:

1. Создайте базу данных test.
2. В ней создайте таблицу users.
3. В этой таблице сделайте 4 поля (столбца):

- **id** (для него нужно поставить галочку AUTO_INCREMENT или A.I.), тип integer,
- **name**, тип varchar, 32 символа,
- **age**, тип integer,
- **birthday** (англ. день рождения), тип date.

4. Найдите вкладку 'вставить' и с ее помощью вставьте несколько строк в эту таблицу При этом поле id заполнять не нужно! Оно заполнится автоматически!
5. Поредактируйте какую-нибудь запись.
6. Удалите какую-нибудь запись.
7. Поменяйте кодировку для таблицы и для всей базы данных (на utf8_general_ci).
8. Переименуйте таблицу.
9. Переименуйте базу данных.

AUTO_INCREMENT

Обратите внимание на то, что мы создали поле **id** и поставили ему галочку **AUTO_INCREMENT**. Это очень важный шаг!

Теперь при вставке новой записи (строки) в таблицу это **поле будет заполняться автоматически**уникальным номером.

При этом если мы удалим строку с каким-то id (например 1), то такого id **больше никогда не будет**.

Зачем нужно поле id?

Затем, чтобы мы всегда могли обратиться к конкретной строке по ее **id**, например с целью удаления или редактирования.

Типы переменных

В **SQL** довольно много типов переменных, но чаще всего приходится пользоваться следующими:

- **integer** – целочисленный.
- **text** – большое текстовое поле.
- **varchar** – не очень большое текстовое поле, при этом мы должны задать его размер (он должен быть степенью двойки: 8, 16, 32, 64, 128, 256 и т.д.).
- **date** - поле для хранения даты (дата хранится в SQL-формате: год-месяц-день, пример: 2013-06-24).

Как работать с mySQL через PHP

Работа с БД из PHP осуществляется всего лишь с помощью трех функций:

- **mysqli_connect** – соединение с сервером и базой данных.
- **mysqli_query** - универсальная функция отправки запросов к БД, с помощью нее можно сделать все.
- **mysqli_error** - вывод ошибок.

Далее подробнее про каждую из функций.

Устанавливаем соединение с БД

Прежде, чем начать работать с базой данных из PHP, нужно **установить соединение с сервером**, на котором эта база находится.

Делается это с помощью функции PHP **mysql_connect**, которая принимает 3 параметра: **имя хоста**(сервера), **имя пользователя**, под которым мы работаем с базой и **пароль** для этого пользователя.

Если вы работаете на своем компьютере то это будут **'localhost'**, **'root'** и пароль в виде пустой строки (на некоторых серверах он тоже может быть **root**). Если ваша база данных **в интернете** - то эти данные дает вам **хостер**.

Давайте установим соединение с базой данных:

```
<?php

        //Устанавливаем доступы к базе данных:

                $host = 'localhost'; //имя хоста, на локальном компьютере это
localhost
```

```
        $user = 'root'; //имя пользователя, по умолчанию это root
        $password = ''; //пароль, по умолчанию пустой
        $db_name = 'test'; //имя базы данных

    //Соединяемся с базой данных используя наши доступы:
        $link = mysqli_connect($host, $user, $password, $db_name);

    /*
        Соединение записывается в переменную $link,
        которая используется дальше для работы mysqi_query.
    */
?>
```

Посылаем запросы к базе данных

Запросы к базе данных представляют собой обычные строки, которые мы вставляем в функцию PHP **mysqli_query** (первым параметром это функция принимает переменную, в которую мы записали результат mysqli_connect, в нашем случае это переменная $link):

```
<?php
    //Внутри функции PHP mysqli_query стоит обычная строка:
    $link = mysqli_query($link, "SELECT*FROM workers WHERE id>0");

    //Мы можем формировать эту строку с помощью переменных:
    $table = 'workers'; //задаем имя таблицы в переменной
    mysqli_query($link, "SELECT*FROM ".$table." WHERE id>0");
?>
```

Принято правило, по которому *команды SQL следует писать в верхнем регистре* (то есть большими буквами), а все остальное - в нижнем.

Это относиться к командам **SELECT, UPDATE, FROM, DELETE, WHERE** и другим такого рода.

Конечно, синтаксической **ошибки не будет**, если вы напишите их маленькими буквами, но принято большими.

Отлавливаем ошибки базы данных

Многие начинающие зачастую не умеют **отлавливать ошибки**, которые вернула база данных.

Поэтому при работе с БД у них постоянно возникают сложности. Что-то не работает, а что - не понятно, так как ошибок они не видят, так как PHP не выводит ошибки mySQL, если ему об этом не сказать

Чтобы вывести ошибки, следует пользоваться конструкцией **or die (mysqli_error($link))**, которую необходимо добавлять к каждому запросу к БД.

Смотрите пример: **mysqli_query($link, $query) or die(mysqli_error($link));**

Таким образом вы сразу будете получать сообщения об ошибках синтаксиса **SQL**. Обратите внимание на то, что на рабочем сайте эти конструкции следует удалять, чтобы пользователи и тем более хакеры не видели ошибок БД.

Проблемы с кодировками

Зачастую у новичков возникает **проблема с кодировками** - казалось бы нормальный русский текст в базу данных записывается абракадаброй или вопросиками.

Чтобы не было проблем с кодировками следует придерживаться простых правил:

- Базу данных следует создавать в кодировке utf8_general_ci.
- Документ PHP должен быть в кодировке utf8.
- Таблицы в БД должны быть в utf8_general_ci.
- На всякий случай сразу после команды mysqli_connect добавьте такое запрос: mysqli_query($link, "SET NAMES 'utf8'");

Начнем практиковаться

Сейчас мы с вами начнем изучить SQL запросы на практике. Для этого нам понадобится тестовая таблица в базе данных, заполненная некоторыми данными. Сейчас мы с вами ее сделаем и заполним.

Итак, создайте свою первую базу данных с помощью PhpMyAdmin.

Назовите ее **"test"**.

Создайте в этой базе новую таблицу.

Назовите ее **"workers"** (англ. работники).

В ней создайте 4 столбца (столбцы по другому называются **поля**):

- **id** – тип integer, не забудьте поставить ему галочку AUTO_INCREMENT (чтобы в этом столбце номера проставлялись автоматически).
- **name** (англ. имя) – тип varchar, размером в 256 знаков.
- **age** (англ. возраст) - тип integer.
- **salary** (англ. зарплата) - тип integer.

Ее заполните тестовыми данными, как показано в таблице ниже (этот шаг обязателен, так как дальше все задачи будут по этой таблице):

id	name	age	salary
1	Дима	23	400
2	Петя	25	500
3	Вася	23	500

4	Коля	30	1000
5	Иван	27	500
6	Кирилл	28	1000

Итак, у нас есть таблица с работниками фирмы, в которой указаны их имена, возрасты и зарплаты (в $). Далее мы будем работать с этой таблицей.

Тестируем работоспособность

Прежде чем начать работу, следует протестировать работоспособность - попробуем сделать какой-нибудь запрос к нашей базе.

Просто скопируйте этот код и запустите его у себя:

```php
<?php
	//Устанавливаем доступы к базе данных:
		$host  =  'localhost'; //имя хоста, на локальном компьютере это localhost
		$user = 'root'; //имя пользователя, по умолчанию это root
		$password = ''; //пароль, по умолчанию пустой
		$db_name = 'test'; //имя базы данных

	//Соединяемся с базой данных используя наши доступы:
		$link = mysqli_connect($host, $user, $password, $db_name);

	//Устанавливаем кодировку (не обязательно, но поможет избежать проблем):
		mysqli_query($link, "SET NAMES 'utf8'");

	//Формируем тестовый запрос:
		$query = "SELECT * FROM workers WHERE id > 0";

	//Делаем запрос к БД, результат запроса пишем в $result:
		$result = mysqli_query($link, $query) or die(mysqli_error($link));

	//Проверяем что же нам отдала база данных, если null - то какие-то проблемы:
		var_dump($result);
?>
```

Если **var_dump($result)** вернет ~~resource, то все работает, если же **null** – то возникли какие-то проблемы.~~ *в новых версиях PHP в **$result** будет лежать объект с данными (всегда будет непустой). Если обработать его через mysqli_fetch_assoc будет или*

результат или null . В таком случае проверьте все еще раз, уберите последовательно все ошибки **PHP**, если таковые есть.

Как достать результат

После того, как мы сделали запрос к базе, в переменной **$result** будет лежать результат этого действия.

Однако лежит он не в той форме, которая нам нужна в PHP, а в той форме, в которой его прислала нам база.

Достать результат в нормальном виде (в массиве) можно с помощью следующего кода:

```
<?php
        //Делаем запрос к БД, результат запроса пишем в $result:
                $result = mysqli_query($link, $query) or die(mysqli_error($link));


        //Преобразуем то, что отдала нам база в нормальный массив PHP $data:
                for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);
?>
```

Как работает последняя строка?

Функция **mysqli_fetch_assoc** считывает последовательно каждую строку результата, который прислала нам база.

В цикле **for** мы считываем построчно результат из базы.

Когда цикл дойдет до последней строки - **mysqli_fetch_assoc** вернет **false** и цикл **for** закончит свою работу.

А результат из БД будет лежать в нормальном виде в массиве **$data**.

Основы работы с базами данных SQL в PHP

Команда SELECT - выборка из БД

Разберемся теперь с командой **SELECT**, которую вы уже видели в тестовом коде.

Команда **SELECT** позволяет нам **достать несколько строк** из нашей таблицы.

Синтаксис ее таков:

```
//ВЫБРАТЬ все_столбцы ИЗ таблицы_такой_то ГДЕ условие_такое_то
SELECT * FROM имя_таблицы WHERE условие_по_которому_следует_выбрать_строки;
```

Далее я привожу **полный код примера**, то есть весь код, который должен быть у вас в документе. Часть этого кода вы уже набрали, когда тестировали работу с базой.

Итак, пример выборки из БД по какому-либо условию, в нашем случае это условие id>0, то есть *выбрать все записи, id которых больше нуля*:

```
<?php
	//Устанавливаем доступы к базе данных:
		$host = 'localhost'; //имя хоста, на локальном компьютере это localhost
		$user = 'root'; //имя пользователя, по умолчанию это root
		$password = ''; //пароль, по умолчанию пустой
		$db_name = 'test'; //имя базы данных

	//Соединяемся с базой данных используя наши доступы:
		mysqli_connect($host, $user, $password, $db_name) or die(mysqli_error($link));

	//Устанавливаем кодировку (не обязательно, но поможет избежать проблем):
		mysqli_query($link, "SET NAMES 'utf8'")
	//ВЫБРАТЬ все_столбцы ИЗ workers ГДЕ ад_ди_больше_нуля (т.е. все)
		$query = "SELECT * FROM workers WHERE id > 0";

	//Делаем запрос к БД, результат запроса пишем в $result:
		$result = mysqli_query($link, $query) or die( mysqli_error($link) );

	//Преобразуем то, что отдала нам база в нормальный массив PHP $data:
		for ($data = []; $row = mysqli_fetch_assoc($result); $data[] = $row);

	//Массив результата лежит в $data, выведем его на экран:
		var_dump($data);
?>
```

Запрос мы делаем к нашей таблице **workers**.

Функция **var_dump** покажет нам, что в массиве **$data** лежат все строки нашей таблицы.

Впредь я не буду приводить полный код, а буду писать только содержимое переменной **$query** (то есть текст запроса).

Что можно писать в условие?

Привожу минимум необходимых команд, на самом деле их больше, но о них ниже: **=** *равно*, **!=** *не равно*, **<>** так тоже *не равно*, **>** *больше*, **<** *меньше*, **>=** *больше либо равно*, **<=** *меньше либо равно*.

Примеры работы с SELECT

Давайте разберем примеры более сложных запросов с **SELECT**.

Выберем из нашей тестовой таблицы workers работника с **id равным 2**:

```
<?php
        //В $data будет лежать одна строка с данными на Петю:
        $query = "SELECT * FROM workers WHERE id=2";
?>
```

Выберем из нашей таблицы workers работников с **id больше 2**:

```
<?php
        //В $data будут все работники, кроме первого и второго:
        $query = "SELECT * FROM workers WHERE id>2";
?>
```

Выберем из нашей таблицы workers работников с **id больше 2 включительно**:

```
<?php
        //В $data будут все работники, кроме первого:
        $query = "SELECT * FROM workers WHERE id>=2";
?>
```

Выберем из нашей таблицы workers работников с **id, не равным 2**:

```
<?php
        //В $data будут все работники кроме Пети (потому что у него id равен 2):
        $query = "SELECT * FROM workers WHERE id != 2";
?>
```

Выберем из нашей таблицы workers работников **возрастом 23 года**:

```
<?php
        //В $data будут Дима и Вася:
        $query = "SELECT * FROM workers WHERE age=23";
?>
```

Выберем из нашей таблицы workers работников с **зарплатой 500$**:

```
<?php
        //В $data будут Петя, Вася и Иван:
        $query = "SELECT * FROM workers WHERE salary=500";
?>
```

Выберем из нашей таблицы workers работника с **именем Дима**:

```
<?php
        //В $data будет только Дима:
        $query = "SELECT * FROM workers WHERE name='Дима'";
```

```
        /*
                Обратите внимание на кавычки:
                поскольку вся строка запроса у нас лежит в двойных
                кавычках, то строку 'Дима' мы возьмем в одинарные!
        */
?>
```

Более сложная логика: OR и AND

В условии выборки можно делать более сложные комбинации с помощью слов **OR** (или) и **AND** (и).

Работают они так же, как и их аналоги в конструкции if.

Примеры на OR и AND

Выберем из таблицы workers работников с **зарплатой 500$ И возрастом 23 года**:

```
<?php
        //В $data будет только Вася (только он подпадает под два условия сразу)
        $query = "SELECT * FROM workers WHERE salary=500 AND age=23";
?>
```

Выберем из таблицы workers работников с **зарплатой 500$ ИЛИ возрастом 23 года**:

```
<?php
        /*
                В $data будут все работники у кого з/п 500$
                и все у кого возраст 23 года.

                Это Дима (23 года), Вася (оба условия),
                Петя (з/п 500$) и Иван (з/п 500$).
        */
        $query = "SELECT * FROM workers WHERE salary=500 OR age=23";
?>
```

Выберем из таблицы workers работников с **зарплатой от 450$ до 900$**:

```
<?php
        //В $data будут Петя (з/п 500$), Вася (з/п 500$), Иван (з/п 500$):
        $query = "SELECT * FROM workers WHERE salary>450 AND salary<900";
?>
```

Выберем из таблицы workers работников с **возрастом от 23 до 27 лет включительно**:

```
<?php
        /*
```

```
                В $data будут Дима (23 года), Петя (25 лет),
                Вася (23 года), Иван (25 лет),
                а вот Коля (30 лет) и Кирилл (28 лет) сюда не попадут:
        */
        $query = "SELECT * FROM workers WHERE age>=23 AND age<=27";
?>
```

Группировки условий

Если вам нужны сложные комбинации команд **OR** и **AND** то, так же, как в **if**, их можно группировать с помощью круглых скобок **( )**, чтобы показать приоритет условий.

Давайте выберем всех работников в возрасте **от 20 лет до 27 лет или с зарплатой от 300$**(обратите внимание на расстановку скобок () - именно они группируют условия):

```
SELECT * FROM workers WHERE (age>20 AND age<27) OR (salary>300)
```

Выбор столбцов

В результат, который вернет нам база данных, не обязательно включать все столбцы.

Пусть мы хотим узнать только имя и возраст работника, не вытягивая из базы данных всю остальную информацию:

```
<?php
        /*
                В $data будут только ключи name и age и не будет id, salary
                (звездочка *, которую мы ставили раньше, командует вернуть ВСЕ
столбцы):
        */
        $query = "SELECT name, age FROM workers WHERE id>0";
?>
```

Команда INSERT - вставка данных в БД

С помощью команды **INSERT** можно добавлять новую информацию в таблицу.

Синтаксис такой:

```
//ВСТАВИТЬ В имя_таблицы УСТАНОВИТЬ поле1=значение1, поле2=значение2, поле3=значение3
INSERT INTO имя_таблицы SET поле1=значение1, поле2=значение2, поле3=значение3...;
```

Давайте добавим нового работника **Гену, в возрасте 30 лет, с зарплатой 1000$**:

```
<?php
        //ВСТАВИТЬ В имя_таблицы УСТАНОВИТЬ имя='Гена', возраст=30, зарплата=1000
        $query = "INSERT INTO workers SET name='Гена', age=30, salary=1000";

        /*
```

```
        Обратите внимание на то, что строку 'Петя'  мы взяли в кавычки.
        Также обратите внимание на то, что поле id мы не указываем -
        база данных сама поставит в него нужное значение.
    */
?>
```

Результат данного действия нужно смотреть в PhpMyAdmin. Не забудьте обновить страницу с таблицей.

```
<?php
    //В $data будет только Вася (только он подпадает под два условия сразу)
    $query = "SELECT * FROM workers WHERE salary=500 AND age=23";
?>
```

Что будет, если не указать значение какого-либо столбца?

```
<?php
    //ВСТАВИТЬ В имя_таблицы УСТАНОВИТЬ имя='Гена', зарплата=1000
    $query = "INSERT INTO workers SET name='Гена', salary=1000";

    //Мы не выставили поле "возраст", хотя оно есть в таблице!
?>
```

Ошибки в данном случае не будет. База данных просто поставит значение по умолчанию для данной колонки. В нашем случае это будет ноль. Проверьте это!

Другой синтаксис INSERT

INSERT имеет альтернативный синтаксис:

```
<?php
    //ВСТАВИТЬ В имя_таблицы (поле1, поле2...) ЗНАЧЕНИЯ (значение1, значение2...)
    $query = "INSERT INTO workers (поле1, поле2...) VALUES (значение1, значение2...)";
?>
```

Давайте добавим нового работника **Гену, в возрасте 30 лет, с зарплатой 1000$** с помощью альтернативного синтаксиса:

```
<?php
    //ВСТАВИТЬ В workers (name, age, salary) ЗНАЧЕНИЯ ('Гена', 30, 1000)"
    $query = "INSERT INTO workers (name, age, salary) VALUES ('Гена', 30, 1000)";
?>
```

Массовая вставка через INSERT

С помощью **INSERT** можно вставлять не одну запись, а несколько. Синтаксис такой:

```
<?php
```

```
        /*
                ВСТАВИТЬ В имя_таблицы (поле1, поле2...)
                        ЗНАЧЕНИЯ (значение1, значение2...), (значение1,
значение2...)...
        */
        $query = "INSERT INTO имя_таблицы (поле1, поле2...)
                VALUES (значение1, значение2...), (значение1, значение2...)...";
?>
```

Давайте добавим одновременно трех новых работников: Гену, в возрасте 30 лет, с зарплатой 1000$, Васю, в возрасте 25 лет, с зарплатой 500$, Ивана, в возрасте 27 лет, с зарплатой 1500$:

```
<?php
        /*
                ВСТАВИТЬ В workers (name, age, salary)
                        ЗНАЧЕНИЯ ('Гена', 30, 1000), ('Вася', 25, 500), ('Иван', 27,
1500)
        */
        $query = "INSERT INTO workers (name, age, salary)
                VALUES ('Гена', 30, 1000), ('Вася', 25, 500), ('Иван', 27, 1500)";
?>
```

Команда DELETE - удаление записей

С помощью команды **DELETE** можно удалять записи из таблицы.

Ее синтаксис очень похож на команду **SELECT**:

```
<?php
        //УДАЛИТЬ ИЗ таблицы_такой_то ГДЕ условие_такое_то
        "DELETE FROM имя_таблицы WHERE условие_по_которому_следует_удалять_строки";
?>
```

Давайте удалим запись с id=6:

```
<?php
        //УДАЛИТЬ ИЗ workers ГДЕ id=6
        $query = "DELETE FROM workers WHERE id=6";

        /*
                Обратите внимание на то,
                что после удаления id=6 в таблице не будет вообще.
        */
?>
```

Задачи на основы работы с базами данных SQL в PHP

Таблица для задач

Все задачи будут по данной таблице **workers** (если не сказано иное):

id	name	age	salary
1	Дима	23	400
2	Петя	25	500
3	Вася	23	500
4	Коля	30	1000
5	Иван	27	500
6	Кирилл	28	1000

Примеры решения задач

Задача

Задача. Выбрать работника с id=10.

Решение:

```
<?php
        $query = "SELECT * FROM workers WHERE id=10";
?>
```

Задача

Задача. Выбрать работников с зарплатой 500$.

Решение:

```
<?php
        $query = "SELECT * FROM workers WHERE salary=500";
?>
```

Задача

Задача. Выбрать работников с зарплатой 500$ и id больше 3.

Решение:

```
<?php
        $query = "SELECT * FROM workers WHERE salary=500 AND id>3";
?>
```

Задача

Задача. Добавьте нового работника Джона, 20 лет, зарплата 700$.

Решение:

Воспользуемся первым синтаксисом:

```
<?php
        $query = "INSERT INTO workers SET name='Джон', age=20, salary=700";
?>
```

Воспользуемся вторым синтаксисом:

```
<?php
        $query = "INSERT INTO workers (name, age, salary) VALUES ('Джон', 20, 700)";
?>
```

Задача

Задача. Добавьте одним запросом трех новых работников: Катю, 20 лет, зарплата 500$, Юлю, 25 лет, зарплата 600$, Женю, 30 лет, зарплата 900$.

Решение: запрос должен выглядеть так:

```
INSERT INTO workers (name, age, salary)
        VALUES ('Катя', 20, 500), ('Юля', 25, 600), ('Женя', 30, 900)
```

Задача

Задача. Удалите работника Джона.

Решение:

```
<?php
```

```
        $query = "DELETE FROM workers WHERE name='Джон'";
?>
```

Задача

Задача. Поставьте Диме зарплату в 1000$.

Решение:

```
<?php
        $query = "UPDATE workers SET salary=1000 WHERE name='Дима'";
?>
```

Задача

Задача. Поставьте Диме зарплату в 1000$ и возраст 20 лет.

Решение:

```
<?php
        $query = "UPDATE workers SET salary=1000, age=20 WHERE name='Дима'";
?>
```

Задачи для решения

На SELECT

Для решения задач данного блока вам понадобятся следующие SQL команды: SELECT, WHERE.

1. Выбрать работника с id = 3.
2. Выбрать работников с зарплатой 1000$.
3. Выбрать работников в возрасте 23 года.
4. Выбрать работников с зарплатой более 400$.
5. Выбрать работников с зарплатой равной или большей 500$.
6. Выбрать работников с зарплатой НЕ равной 500$.
7. Выбрать работников с зарплатой равной или меньшей 900$.
8. Узнайте зарплату и возраст Васи.

На OR и AND

Для решения задач данного блока вам понадобятся следующие SQL команды: SELECT, WHERE, OR, AND.

9. Выбрать работников в возрасте от 25 (не включительно) до 28 лет (включительно).

10. Выбрать работника Петю.
11. Выбрать работников Петю и Васю.
12. Выбрать всех, кроме работника Петя.
13. Выбрать всех работников в возрасте 27 лет или с зарплатой 1000$.
14. Выбрать всех работников в возрасте от 23 лет (включительно) до 27 лет (не включительно) или с зарплатой 1000$.
15. Выбрать всех работников в возрасте от 23 лет до 27 лет или с зарплатой от 400$ до 1000$.
16. Выбрать всех работников в возрасте 27 лет или с зарплатой не равной 400$.

На INSERT

Для решения задач данного блока вам понадобятся следующие SQL команды: INSERT.

17. Добавьте нового работника Никиту, 26 лет, зарплата 300$. Воспользуйтесь **первым** синтаксисом.
18. Добавьте нового работника Светлану с зарплатой 1200$. Воспользуйтесь **вторым** синтаксисом.
19. Добавьте двух новых работников одним запросом: Ярослава с зарплатой 1200$ и возрастом 30, Петра с зарплатой 1000$ и возрастом 31.

На DELETE

Для решения задач данного блока вам понадобятся следующие SQL команды: DELETE.

20. Удалите работника с id=7.
21. Удалите Колю.
22. Удалите всех работников, у которых возраст 23 года.

Верните таблицу workers в исходное состояние.

На UPDATE

Для решения задач данного блока вам понадобятся следующие SQL команды: UPDATE.

23. Поставьте Васе зарплату в **200$**.
24. Работнику с id=4 поставьте возраст **35** лет.
25. Всем, у кого зарплата 500$ сделайте ее **700$**.
26. Работникам с id больше 2 и меньше 5 включительно поставьте возраст **23**.
27. Поменяйте Васю на Женю и прибавьте ему зарплату до **900$**.