

**ВИДЕОКУРС Основы Баз данных от Geek Brains (20 уроков) <https://www.mysql.com/>**

Эта шпаргалка будет бесполезна без просмотра видеокурса

<https://geekbrains.ru/courses/86>

**СОДЕРЖАНИЕ (файл состоит из нескольких листов)**

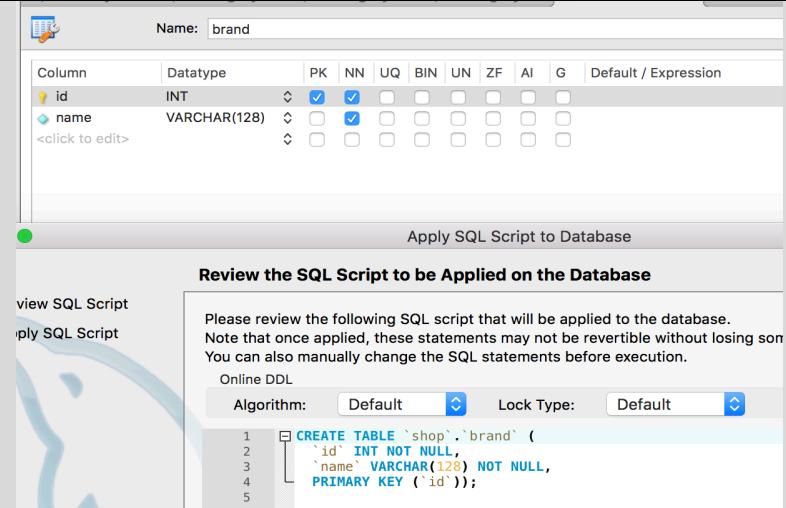
Lesson 5	Создание (CREATE) и заполнение (INSERT) таблиц базы данных
Lesson 6	SQL-команды SELECT и WHERE для вывода данных из базы/ SQL commands SELECT and WHERE
Lesson 7	Вывод данных с условиями и сортировкой (SQL-команды DISTINCT, ORDER BY, LIMIT)
Lesson 8	Изменение\Удаление данных SQL-команды DELETE и UPDATE
Lesson 9	Согласованность \ Консистентность. Создаем\заполняем таблицу Products Настройка согласованности данных ВНЕШНИЙ КЛЮЧ - FOREIGN KEY constraints. Связываем id_brand в таб.
Lesson 10	Products со столбцом id в таб. Brand. ВИЗУАЛИЗАЦИЯ связей в таблицах Database для хранения данных заказанных продуктов.Настройка связей в таблице ORDER_PRODUCTS с PRODUCTS и
Lesson 11	ORDER Составной первичный ключ. Создаем таблицу ORDER_PRODUCTS - связывающая таблица заказов из ORDER с
Lesson 12	товарами из PRODUCTS
Lesson 13	ТЕОРИЯ. Объединение данных из нескольких таблиц
Lesson 14	Объединение данных из нескольких таблиц (INNER JOIN)
Lesson 15	Объединение данных из нескольких таблиц (LEFT JOIN)
Lesson 16	ПОЛНОЕ объединение данных из НЕСКОЛЬКИХ таблиц (FULL JOIN FULL OUTER JOIN). UNION
Lesson 17	Агрегирующие функции (count, sum, max, min) считаем сумму в заказе у Васи Оператор GROUP BY. Объединение по заказчику, вывод итого суммы по заказу, итого количество товаро в заказе
Lesson 18	(min, max, sum...)
Lesson 19	Ограничения для агрегированных данных HAVING. ИНДЕКСЫ для оптимизации работы DataBase
Lesson 20	Транзакции, на примере банковского перевода от одного Usera другому

Присоединяйтесь ко мне в Instagram, делюсь наглядными инструкциями и шпаргалками по курсам, которые я прошла на Geek Brains

[https://www.instagram.com/natalya\\_dol\\_frontender/](https://www.instagram.com/natalya_dol_frontender/)

## Lesson 5

## Команды на языке SQL, ЗАПОЛНЯЕМ ТАБЛИЦЫ

№	Действие	Команда на языке SQL и скрин результата
1	CREATE the Tables. Table - Create table	<p><b>CREATE TABLE `shop`.`brand` (</b></p> <p><b>  `id` INT NOT NULL,</b></p> <p><b>  `name` VARCHAR(128) NOT NULL,</b></p> <p><b>  PRIMARY KEY (`id`));</b></p> 

2

Заполнение  
таблицы

The screenshot shows a MySQL Workbench interface. At the top is a toolbar with 'Result Grid' selected, followed by 'Edit' and 'Export/Import' buttons. Below the toolbar is a result grid table:

ID	name	discount
1	Женская одежда	5
2	Мужская одежда	0
HULL	HULL	HULL

Below the table is a button labeled 'Apply SQL Script to Database'. A modal window titled 'Review the SQL Script to be Applied on the Database' contains the following text and code:

Please review the following SQL script that will be applied to the database.  
Note that once applied, these statements may not be revertible without losing some of the data.  
You can also manually change the SQL statements before execution.

```
1 INSERT INTO `shop`.`category` (`ID`, `name`, `discount`) VALUES ('1', 'Женская одежда', '5');
2 INSERT INTO `shop`.`category` (`ID`, `name`) VALUES ('2', 'Мужская одежда');
```

At the bottom of the modal window, the SQL statements are displayed in red:

**INSERT INTO `shop`.`category` (`ID`, `name`, `discount`) VALUES ('1', 'Женская  
одежда', '5');**  
**INSERT INTO `shop`.`category` (`ID`, `name`) VALUES ('2', 'Мужская одежда');**

3  
Заполнение  
таблицы через код  
SQL

The screenshot shows the MySQL Workbench interface. At the top, there are two tabs: 'category' (with a red '2') and 'category' (with a red '1'). Below the tabs is a toolbar with various icons. The main area contains an SQL query editor with the following code:

```
1 • use shop;
2 •
3 • INSERT INTO category (ID, name, discount) VALUES (5, 'Детская обувь', 10);
```

To the right of the editor is a 'Result Grid' window displaying the data from the 'category' table:

id	name	discount
1	Женская одежда	5
2	Мужская одежда	0
3	Женская обувь	10
4	Мужская обувь	15
5	Детская обувь	10
NULL	NULL	NULL

Below the grid, the text 'use shop;' is displayed in red, followed by two additional SQL statements in red:

**use shop;**

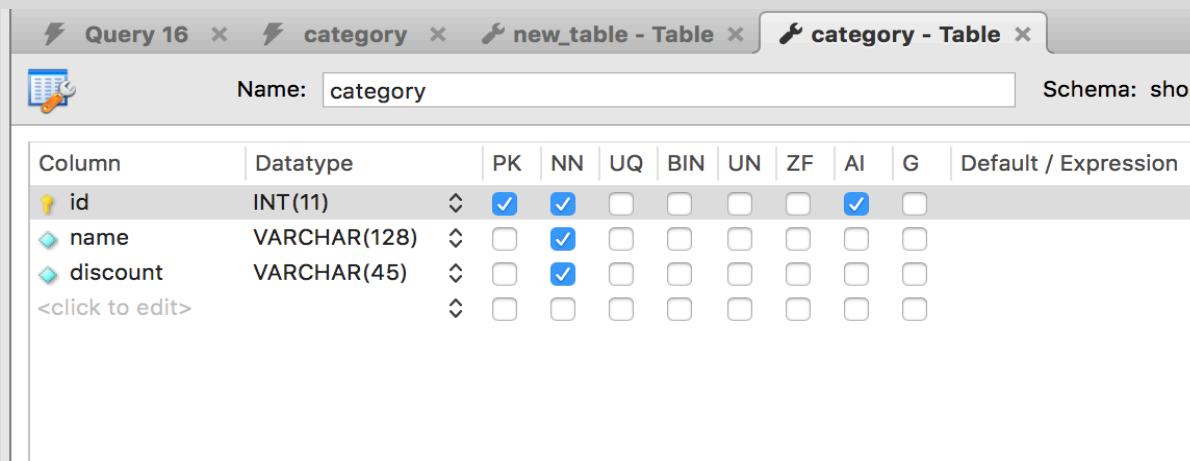
**INSERT INTO category (ID, name, discount) VALUES (3, 'Женская обувь', 10);**

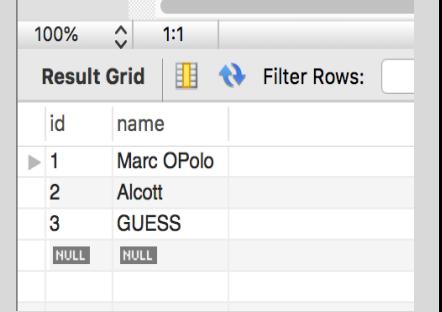
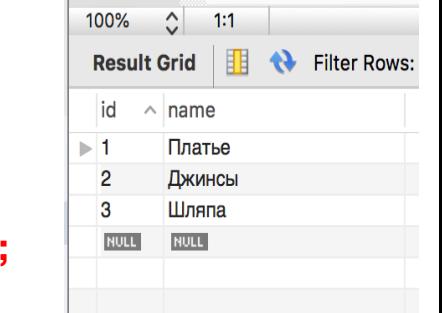
**INSERT INTO category (ID, name, discount) VALUES (4, 'Мужская обувь', 15);**

4  
Если текст с  
ковычками смотри  
'man' 's shoes'

The screenshot shows the MySQL Workbench interface with a single tab labeled 'category'. The main area contains an SQL query editor with the following code:

```
1 • use shop;
2
3 • INSERT INTO category (id, name, discount, alias_name) VALUES (3, 'Женская обувь', 10, NULL);
4 • INSERT INTO category (id, name, discount, alias_name) VALUES (4, 'Мужская обувь', 15, 'man' 's shoes');
```

5 (6:19)	Чтобы не прописывать каждый раз ID	<p><b>при создании таблицы нужно поставить галочку в AI в строке ID. Кликаем Alter Table и правим</b></p>  <pre>ALTER TABLE `shop`.`category` CHANGE COLUMN `id` `id` INT(11) NOT NULL AUTO_INCREMENT ;</pre>
6	Вставить строку в таблицу теперь можно так, выполняем команду, нажимаем молнию	<pre>use shop;</pre> <pre>INSERT INTO category ( name, discount) VALUES ('Шляпы', 0);</pre>

7	Заполняем таблицу brand	<pre><code>use shop;  INSERT INTO brand (name) VALUES ('Marc O' 'Polo'); INSERT INTO brand (name) VALUES ('Alcott'); INSERT INTO brand (name) VALUES ('GUESS');</code></pre> 
8	Заполняем таблицу product type	<pre><code>use shop;  INSERT INTO product_type (name) VALUES ('Платье'); INSERT INTO product_type (name) VALUES ('Джинсы'); INSERT INTO product_type (name) VALUES ('Шляпа');</code></pre> 

Образец базы

	A	B	C	D	E	F	G
1	Таблица "Товары"						
2	Артикул	ID бренда	ID типа товара	ID категории	Цена		
3	153921109/874	1	1	1	15980		
4	TS10757DO C109	2	2	1	1090		
5	N61K69-W7950-G75C	3	1	1	13990		
6	TS9682UO C099	2	2	2	1490		
7							
8							
9							
10	Таблица "Категория товаров"				Таблица "Бренд"		
11	ID	Категория	Скидка		ID	Тип	
12	1	Женская одежда	5%		1	Marc O'Polo	
13	2	Мужская одежда	0%		2	ALCOTT	
14					3	GUESS	
15							
16							
17	Таблица "Тип товара"						
18	ID	Тип					
19	1	Платье					
20	2	Футболка					
21							

!!!!!!

Комментарии  
обрамляем в --  
текст --

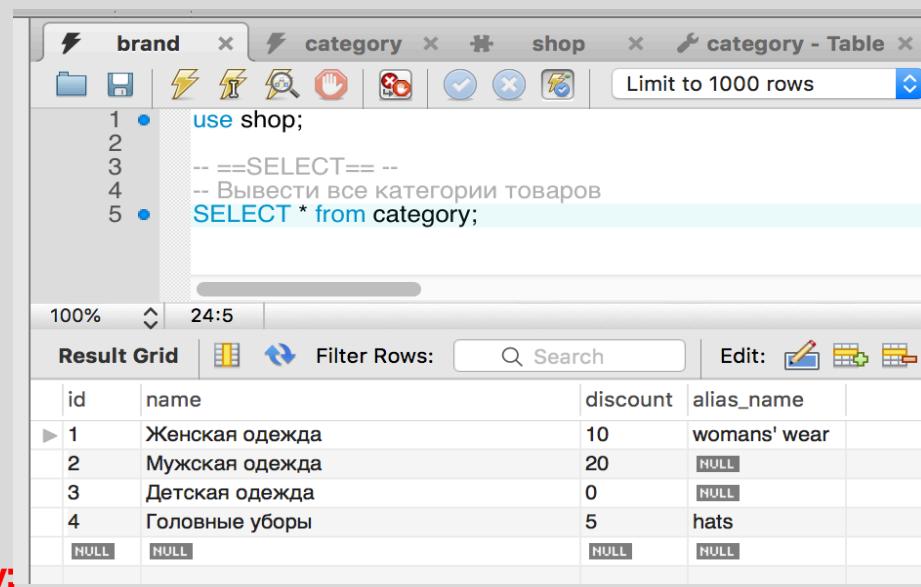
```
-- == SELECT <столбец> == --
-- вывести названия всех категорий товаров
SELECT name FROM category;

-- вывести названия и скидки товаров
SELECT discount, name FROM category;

-- вывести все скидки
SELECT discount FROM category;
```

## Lesson 6

## Урок 6. SQL-команды SELECT и WHERE для вывода данных из базы/ SQL commands SELECT and WHERE

№	Действие	Команда на языке SQL и скрин результата																														
1	All categories Вывести все категории товаров	<p>use shop;</p> <p><b>SELECT * from category;</b></p>  <table border="1"><thead><tr><th></th><th>id</th><th>name</th><th>discount</th><th>alias_name</th></tr></thead><tbody><tr><td>▶</td><td>1</td><td>Женская одежда</td><td>10</td><td>womans' wear</td></tr><tr><td></td><td>2</td><td>Мужская одежда</td><td>20</td><td>NULL</td></tr><tr><td></td><td>3</td><td>Детская одежда</td><td>0</td><td>NULL</td></tr><tr><td></td><td>4</td><td>Головные уборы</td><td>5</td><td>hats</td></tr><tr><td></td><td>NULL</td><td>NULL</td><td>NULL</td><td>NULL</td></tr></tbody></table>		id	name	discount	alias_name	▶	1	Женская одежда	10	womans' wear		2	Мужская одежда	20	NULL		3	Детская одежда	0	NULL		4	Головные уборы	5	hats		NULL	NULL	NULL	NULL
	id	name	discount	alias_name																												
▶	1	Женская одежда	10	womans' wear																												
	2	Мужская одежда	20	NULL																												
	3	Детская одежда	0	NULL																												
	4	Головные уборы	5	hats																												
	NULL	NULL	NULL	NULL																												

2 По значению

The screenshot shows a MySQL Workbench interface. In the top-left pane, there are tabs for 'Query 44' (selected), 'product\_type', 'product\_type', 'brand', and a 'Limit to 1000 rows' dropdown. Below the tabs is a code editor with the following SQL query:

```
1 • use shop;
2
3 -- == SELECT ==
4 -- Вывести все категории товаров с идентификатором, равным 3
5 • SELECT * from category WHERE id = 3;
```

The code editor has line numbers 1 through 7. Lines 1 and 5 are marked with blue dots, indicating they are selected. Lines 3 and 4 are comments. Line 5 is the main query. Below the code editor is a status bar showing '100%' and '1:6'. At the bottom are buttons for 'Result Grid' (selected), 'Edit', and 'Export'.

In the bottom-right pane, there is a 'Result Grid' table with the following data:

	id	name	discount
▶	3	Женская обувь	10
	HULL	HULL	HULL

use shop;  
SELECT \* from category WHERE id = 3;

3  
По условию  
Вывести все  
категории товаров,  
у которых скидка не  
равна 0

use shop;

**SELECT \* from category WHERE discount <> 0;**

The screenshot shows a MySQL Workbench interface. In the top-left pane, there are tabs for 'Query 44' (selected), 'product\_type', 'product\_type', 'brand', and a connection icon. Below the tabs is a toolbar with icons for file operations, search, and other database functions. A dropdown menu says 'Limit to 1000 rows'. The main area contains a code editor with the following SQL query:

```
1 ● use shop;
2
3 -- == SELECT ==
4 -- Вывести все категории товаров, у которых скидка не равна 0
5 ● SELECT * from category WHERE discount <> 0;
```

The code editor has line numbers 1 through 7. The explanatory comments are in Russian. The result grid below shows the data returned by the query:

	id	name	discount
▶	1	Женская одежда	5
	3	Женская обувь	10
	4	Мужская обувь	15
	5	Детская обувь	10
	NULL	NULL	NULL

4  
Вывести все  
категории товаров,  
у которых скидка  
больше 5

use shop;  
SELECT \* from category WHERE discount > 5;

The screenshot shows a MySQL Workbench interface. The query editor tab is titled "Query 44". The code entered is:

```
1 use shop;
2 -- == SELECT ==
3 -- Вывести все категории товаров, у которых скидка больше 5
4
5 SELECT * from category WHERE discount > 5;
```

The result grid shows the following data:

	id	name	discount
▶	3	Женская обувь	10
	4	Мужская обувь	15
	5	Детская обувь	10
	NULL	NULL	NULL

5

Вывести все  
категории товаров,  
у которых скидка  
больше 5 и меньше  
15

use shop;

**SELECT \* from category WHERE (discount > 5) AND (discount < 15);**

The screenshot shows the MySQL Workbench interface. In the top-left pane, there is a list of queries and tables. The main pane displays a query editor with the following code:

```
1 ● use shop;
2
3 -- == SELECT ==
4 -- Вывести все категории товаров, у которых скидка больше 5 и меньше 15
5 ● SELECT * from category WHERE (discount > 5) AND (discount < 15);
6
7
```

Below the code, the results grid shows the following data:

id	name	discount
3	Женская обувь	10
5	Детская обувь	10
NULL	NULL	NULL

6

Вывести все категории товаров, у которых скидка меньше 5 или больше или равна 10

use shop;

**SELECT \* from category WHERE (discount < 5) OR (discount >= 10);**

The screenshot shows the MySQL Workbench interface. In the query editor tab 'Query 44', the following SQL code is written:

```
1 use shop;
2 -- == SELECT ==
3 -- Вывести все категории товаров, у которых скидка меньше 5 или больше и
4 -- OR (discount >= 10);
5
6
7
```

The result grid displays the following data:

	id	name	discount
▶	2	Мужская одежда	0
	3	Женская обувь	10
	4	Мужская обувь	15
	5	Детская обувь	10
	6	Шляпы	0
	7	Шляпы	0
	NULL	NULL	NULL

7

Вывести все  
категории товаров,  
у которых скидка не  
меньше 5

use shop;

**SELECT \* from category WHERE NOT (discount < 5);**

The screenshot shows a MySQL Workbench interface with a query editor and a result grid. The query editor contains the following code:

```
1 • use shop;
2
3 -- == SELECT ==
4 -- Вывести все категории товаров, у которых скидка не меньше 5
5 • SELECT * from category WHERE NOT (discount < 5);
6
```

The result grid displays the following data:

	id	name	discount
1	1	Женская одежда	5
▶	3	Женская обувь	10
	4	Мужская обувь	15
	5	Детская обувь	10
	NULL	NULL	NULL

Вывести тех у кого есть значение в alias\_name

8 Вывести все категории товаров, у которых есть псевдоним

use shop;

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

```
use shop;
-- SELECT --
-- Вывести все категории товаров, у которых есть псевдоним
SELECT * from category WHERE alias_name IS NOT NULL;
```

The result grid displays the following data:

	id	name	discount	alias_name
▶	1	Женская одежда	10	womans' wear
	4	Головные уборы	5	hats
	NULL	NULL	NULL	NULL

**SELECT \* from category WHERE alias\_name IS NOT NULL;**

Вывести тех у кого НЕТ значения в alias\_name

9 Вывести все категории товаров, у которых есть псевдоним

use shop;

The screenshot shows the MySQL Workbench interface with a query editor and a result grid. The query is:

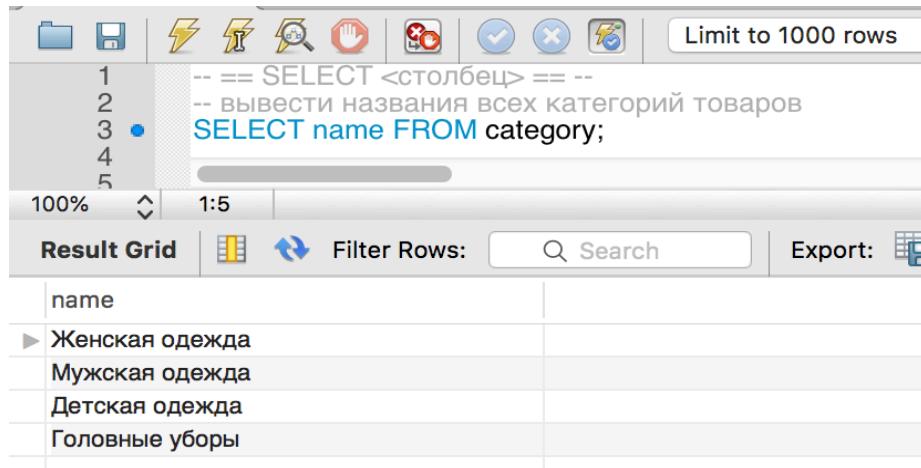
```
use shop;
-- SELECT --
-- Вывести все категории товаров, у которых есть псевдоним
SELECT * from category WHERE alias_name IS NULL;
```

The result grid displays the following data:

	id	name	discount	alias_name
▶	2	Мужская одежда	20	NULL
	3	Детская одежда	0	NULL
	NULL	NULL	NULL	NULL

**SELECT \* from category WHERE alias\_name IS NULL;**

--	--	--

Lesson 7		<h2 style="color: #0070C0;">Урок 7. SQL-команды DISTINCT, ORDER BY, LIMIT</h2> <div style="border: 1px solid #ccc; padding: 5px; float: right;"> <ul style="list-style-type: none"> <li>• Получение данных:</li> <li>• SELECT</li> <li>• WHERE</li> <li>• DISTINCT</li> <li>• ORDER BY (ASC, DESC)</li> <li>• LIMIT</li> </ul> </div>					
№	Действие	<b>Команда на языке SQL и скрин результата</b>					
1	вывести названия всех категорий товаров	 <p>The screenshot shows the MySQL Workbench interface. In the SQL editor, the following query is written:</p> <pre>-- == SELECT &lt;столбец&gt; == -- -- вывести названия всех категорий товаров SELECT name FROM category;</pre> <p>The result grid below displays the following data:</p> <table border="1"> <thead> <tr> <th>name</th> </tr> </thead> <tbody> <tr> <td>Женская одежда</td> </tr> <tr> <td>Мужская одежда</td> </tr> <tr> <td>Детская одежда</td> </tr> <tr> <td>Головные уборы</td> </tr> </tbody> </table> <p>At the bottom of the SQL editor, the command is highlighted in red:</p> <p style="color: red;">use shop; SELECT name FROM category;</p>	name	Женская одежда	Мужская одежда	Детская одежда	Головные уборы
name							
Женская одежда							
Мужская одежда							
Детская одежда							
Головные уборы							

2

вывести названия и скидки товаров

-- вывести названия и скидки товаров  
SELECT discount, name FROM category;

Result Grid | Filter Rows: | Export:

discount	name
10	Женская одежда
20	Мужская одежда
0	Детская одежда
5	Головные уборы

use shop; SELECT discount, name FROM category;

3

вывести все скидки

1 ● SELECT discount FROM category;

Result Grid | Filter Rows: | Export:

discount
10
20
0
5

use shop; SELECT discount FROM category;

1 • SELECT discount FROM category;

2

100% 31:1

Result Grid Filter Rows: Search

discount
10
20
0
5

4 **DISTINCT**  
вывести все  
**уникальные**  
значения скидок

use shop; **SELECT DISTINCT discount FROM category;**

1 • SELECT \* FROM category ORDER BY discount;

100% 41:1

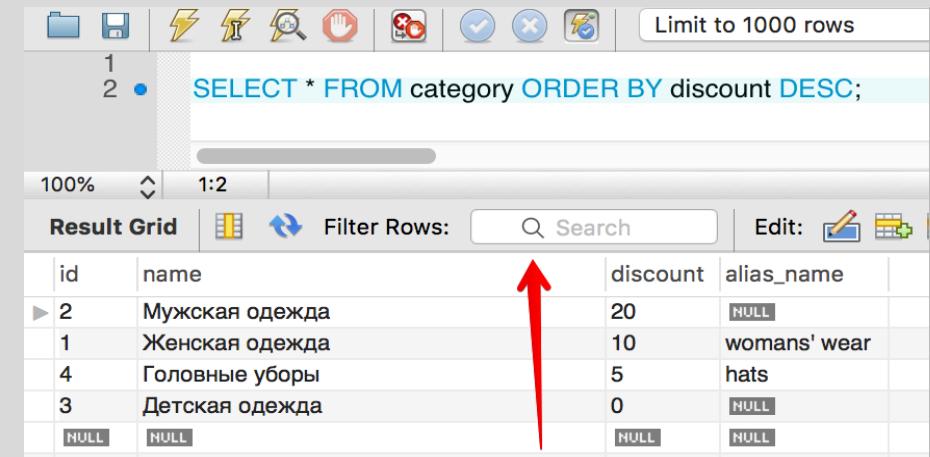
Result Grid Filter Rows: Search Edit:

id	name	discount	alias_name
3	Детская одежда	0	NULL
4	Головные уборы	5	hats
1	Женская одежда	10	womans' wear
2	Мужская одежда	20	NULL
NULL	NULL	NULL	NULL

5 **ORDER BY** вывести  
все категории  
товаров, и  
**отсортировать** их  
по размеру скидки

use shop; **SELECT \* FROM category ORDER BY discount;**

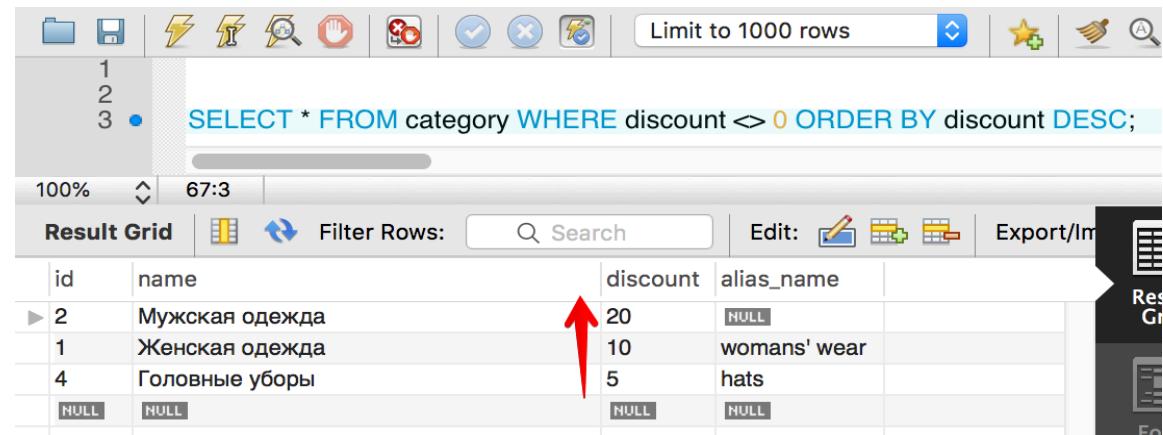
6 ORDER BY  
вывести все  
категории товаров,  
и отсортировать их  
по размеру скидки  
в обратном  
порядке



id	name	discount	alias_name
2	Мужская одежда	20	NULL
1	Женская одежда	10	womans' wear
4	Головные уборы	5	hats
3	Детская одежда	0	NULL
NULL	NULL	NULL	NULL

use shop; SELECT \* FROM category ORDER BY discount DESC;

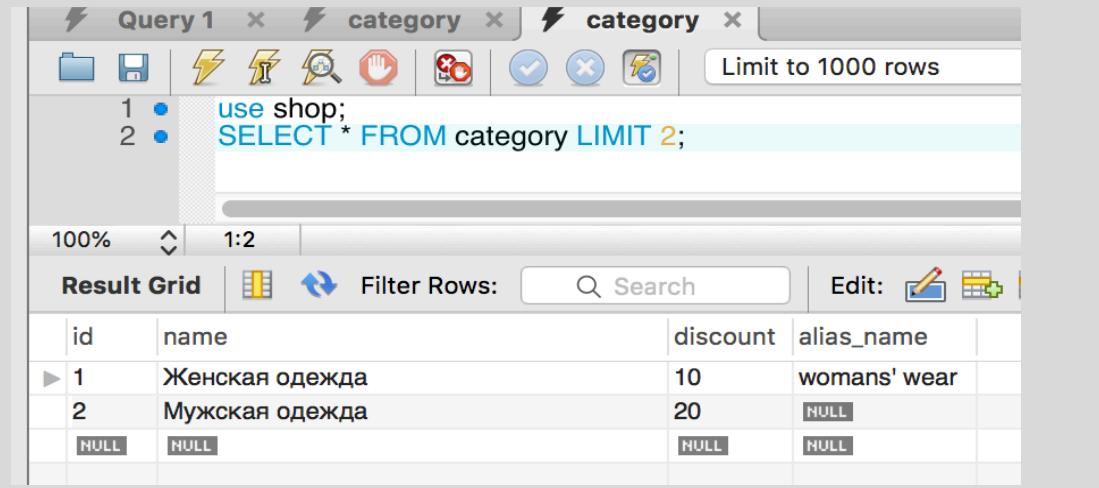
7 вывести все  
категории товаров с  
ненулевой  
скидкой, и  
отсортировать их  
по размеру скидки  
в обратном порядке



id	name	discount	alias_name
2	Мужская одежда	20	NULL
1	Женская одежда	10	womans' wear
4	Головные уборы	5	hats
NULL	NULL	NULL	NULL

use shop; SELECT \* FROM category WHERE discount >> 0 ORDER BY  
discount DESC;

8 LIMIT  
вывести первые 2  
категории товара



The screenshot shows the MySQL Workbench interface with three tabs: 'Query 1', 'category', and 'category'. The 'category' tab is active. The SQL editor contains the following code:

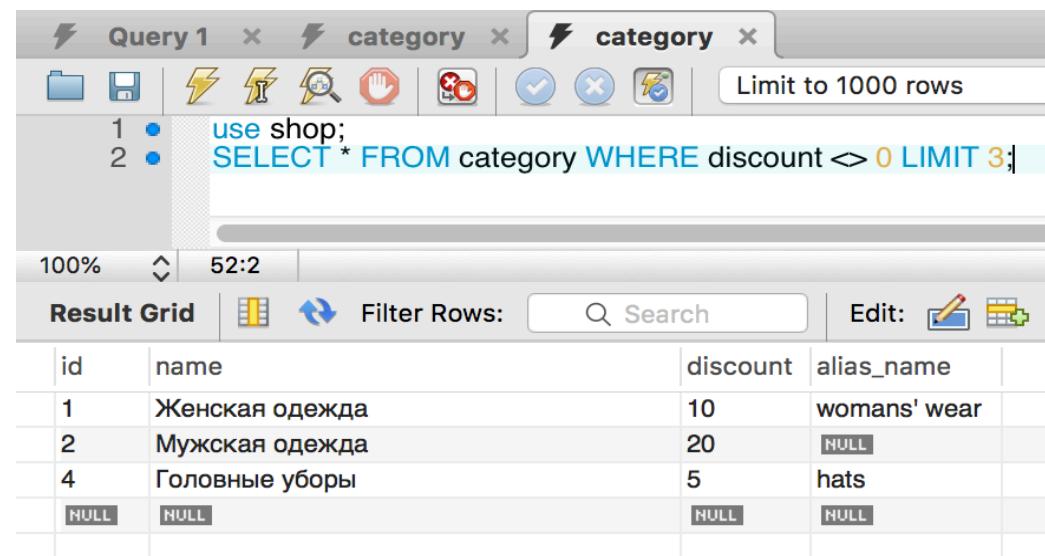
```
use shop;
SELECT * FROM category LIMIT 2;
```

The result grid displays the following data:

	id	name	discount	alias_name
▶	1	Женская одежда	10	womans' wear
	2	Мужская одежда	20	NULL
	NULL	NULL	NULL	NULL

use shop; SELECT \* FROM category LIMIT 2;

9  
вывести первые 3  
категории товара со  
скидкой не равной  
нулю



The screenshot shows the MySQL Workbench interface with three tabs: 'Query 1', 'category', and 'category'. The 'category' tab is active. The SQL editor contains the following code:

```
use shop;
SELECT * FROM category WHERE discount <> 0 LIMIT 3;
```

The result grid displays the following data:

	id	name	discount	alias_name
▶	1	Женская одежда	10	womans' wear
	2	Мужская одежда	20	NULL
	4	Головные уборы	5	hats
	NULL	NULL	NULL	NULL

use shop;  
SELECT \* FROM category WHERE discount <> 0 LIMIT 3;

10

Получить все  
категории товаров  
со скидкой < 10%, и  
отсортировать их по  
названию

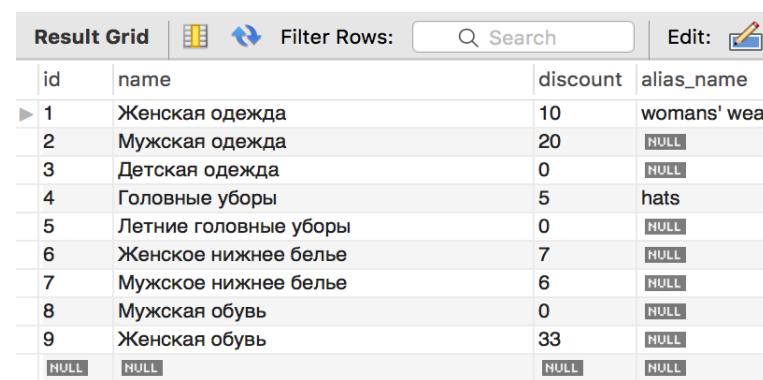
The screenshot shows the MySQL Workbench interface. At the top, there is a toolbar with various icons. Below the toolbar, the SQL editor contains two lines of code:

```
1 • use shop;  
2 • SELECT * FROM category WHERE discount < 10 ORDER BY name;
```

The status bar at the bottom indicates "100%" and "58:2". Below the editor is the "Result Grid" section, which displays the query results in a table format:

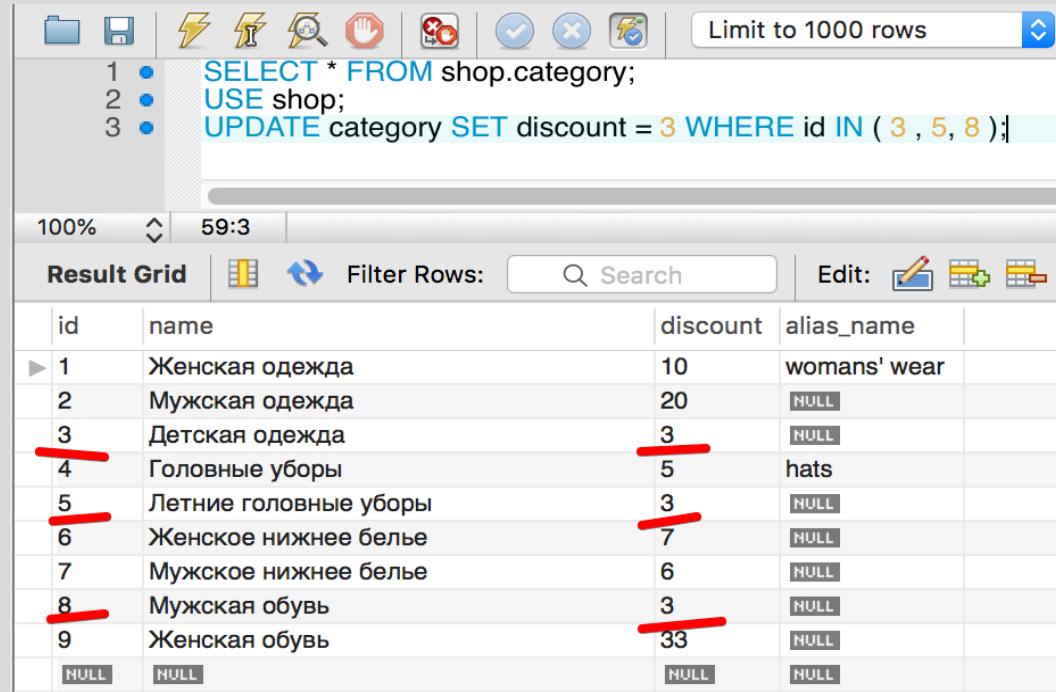
id	name	discount	alias_name
4	Головные уборы	5	hats
3	Детская одежда	0	NULL
6	Женское нижнее белье	7	NULL
8	Мужская обувь	0	NULL
7	Мужское нижнее белье	6	NULL
5	Шляпы	0	NULL

**use shop;  
SELECT \* FROM category WHERE discount < 10 ORDER BY name;**

Lesson 8		<b>Урок 8. SQL-команды DELETE и UPDATE</b>																																												
№	Действие	<b>Команда на языке SQL и скрин результата</b>																																												
1	<u>Изменить name</u> <u>Шляпы на Летние</u> <u>головные уборы по</u> <u>ID (Было ID 5</u> <u>Шляпы, станет ID 5</u> <u>Летние головные</u> <u>уборы)</u>	<p style="color: red;">USE shop;</p> <p style="color: red;">UPDATE category SET name = 'Летние головные уборы' WHERE id = 5;</p>  <table border="1"> <thead> <tr> <th>id</th> <th>name</th> <th>discount</th> <th>alias_name</th> </tr> </thead> <tbody> <tr> <td>1</td> <td>Женская одежда</td> <td>10</td> <td>womans' wear</td> </tr> <tr> <td>2</td> <td>Мужская одежда</td> <td>20</td> <td>NULL</td> </tr> <tr> <td>3</td> <td>Детская одежда</td> <td>0</td> <td>NULL</td> </tr> <tr> <td>4</td> <td>Головные уборы</td> <td>5</td> <td>hats</td> </tr> <tr> <td>5</td> <td>Летние головные уборы</td> <td>0</td> <td>NULL</td> </tr> <tr> <td>6</td> <td>Женское нижнее белье</td> <td>7</td> <td>NULL</td> </tr> <tr> <td>7</td> <td>Мужское нижнее белье</td> <td>6</td> <td>NULL</td> </tr> <tr> <td>8</td> <td>Мужская обувь</td> <td>0</td> <td>NULL</td> </tr> <tr> <td>9</td> <td>Женская обувь</td> <td>33</td> <td>NULL</td> </tr> <tr> <td>NULL</td> <td>NULL</td> <td>NULL</td> <td>NULL</td> </tr> </tbody> </table>	id	name	discount	alias_name	1	Женская одежда	10	womans' wear	2	Мужская одежда	20	NULL	3	Детская одежда	0	NULL	4	Головные уборы	5	hats	5	Летние головные уборы	0	NULL	6	Женское нижнее белье	7	NULL	7	Мужское нижнее белье	6	NULL	8	Мужская обувь	0	NULL	9	Женская обувь	33	NULL	NULL	NULL	NULL	NULL
id	name	discount	alias_name																																											
1	Женская одежда	10	womans' wear																																											
2	Мужская одежда	20	NULL																																											
3	Детская одежда	0	NULL																																											
4	Головные уборы	5	hats																																											
5	Летние головные уборы	0	NULL																																											
6	Женское нижнее белье	7	NULL																																											
7	Мужское нижнее белье	6	NULL																																											
8	Мужская обувь	0	NULL																																											
9	Женская обувь	33	NULL																																											
NULL	NULL	NULL	NULL																																											

2 В ID: 3.Детская  
одежда, 5. Летние  
уборы, 8. Мужская  
обувь - [изменим](#)  
[скидку на 3%](#)

USE shop;  
UPDATE category SET discount = 3 WHERE id IN ( 3 , 5, 8 );



The screenshot shows the MySQL Workbench interface with the following details:

- Toolbar:** Includes icons for file operations (New, Save, Import, Export, Find, Stop, Run, Refresh), a search bar, and a "Limit to 1000 rows" dropdown.
- Query Editor:** Displays three SQL statements:
  - 1 • SELECT \* FROM shop.category;
  - 2 • USE shop;
  - 3 • UPDATE category SET discount = 3 WHERE id IN ( 3 , 5, 8 );
- Result Grid:** Shows the contents of the 'category' table.

id	name	discount	alias_name
1	Женская одежда	10	womans' wear
2	Мужская одежда	20	NULL
3	Детская одежда	3	NULL
4	Головные уборы	5	hats
5	Летние головные уборы	3	NULL
6	Женское нижнее белье	7	NULL
7	Мужское нижнее белье	6	NULL
8	Мужская обувь	3	NULL
9	Женская обувь	33	NULL
NULL	NULL	NULL	NULL
- Bottom Status Bar:** Shows "100%" zoom level and "59:3" time.

- 3** Удалить категорию  
Женская Обувь по  
ID 9

**USE shop;**  
**DELETE FROM category WHERE id = 9;**

The screenshot shows a MySQL Workbench interface. At the top, there is a script editor window containing three lines of SQL code:

```
1 •   SELECT * FROM shop.category;
2 •   USE shop;
3 •   DELETE FROM category WHERE id = 9;
```

Below the script editor is a status bar showing "100%" and "29:1". Underneath the status bar is a toolbar with icons for "Result Grid", "Filter Rows:", "Search", and "Edit".

The main area is a "Result Grid" table with the following data:

	id	name	discount	alias_name
▶	1	Женская одежда	10	womans' wear
	2	Мужская одежда	20	NULL
	3	Детская одежда	3	NULL
	4	Головные уборы	5	hats
	5	Летние головные уборы	3	NULL
	6	Женское нижнее белье	7	NULL
	7	Мужское нижнее белье	6	NULL
	8	Мужская обувь	3	NULL
	NULL	NULL	NULL	NULL

**Таблица "Товары"**

	Артикул	ID бренда	ID типа товара	ID категории	Цена
1	153921109/874	1	1	1	15980
2	TS10757DO C109	2	2	1	1090
3	N61K69-W79S0-G75C	3	1	1	13990
4	TS9682UO C099	2	2	2	1490

**Таблица "Категория товаров"**

ID	Категория	Скидка
1	Женская одежда	5%
2	Мужская одежда	0%

**Таблица "Бренд"**

ID	Тип
1	Marc O'Polo
2	ALCOTT
3	GUESS

**Таблица "Тип товара"**

ID	Тип
1	Платье
2	Футболка

## Консистентность данных

Консистентность данных ([англ. data consistency](#) или [data validity](#)) — это согласованность данных друг с другом, целостность данных, а также внутренняя непротиворечивость. Множество всех условий, налагаемых на данные определяется [моделью](#)

The screenshot shows three separate windows in MySQL Workbench, each displaying a query and its results.

**Query 1 (Top Left):**

```
SELECT * FROM shop.brand;
```

**Result Grid:**

id	name
1	Marc OPolo
2	Alcott
3	GUESS
4	ADIDAS
5	H&M
NONE	NONE

**Query 2 (Top Right):**

```
SELECT * FROM shop.category;  
USE shop;
```

**Result Grid:**

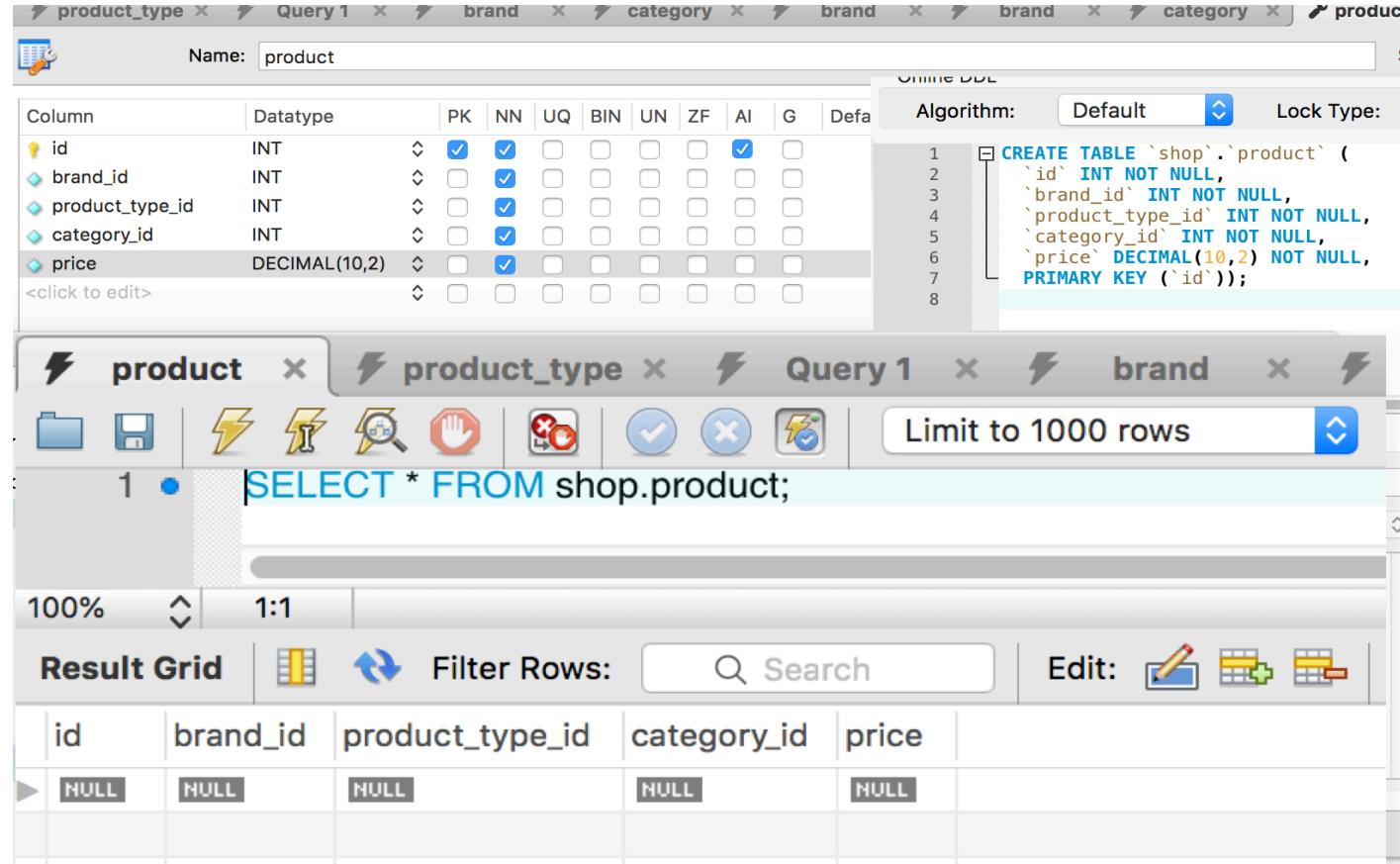
id	name	discount	alias_name
1	Женская одежда	10	women's clothing
2	Мужская одежда	20	men's clothing
3	Детская одежда	3	baby clothes
4	Головные уборы	5	hats
5	Летние головные уборы	3	summer hats
6	Женское нижнее белье	7	women's secret clothing
7	Мужское нижнее белье	6	men's secret clothing
8	Мужская обувь	3	men's shoes
NONE	NONE	NONE	NONE

**Query 3 (Bottom Left):**

```
SELECT * FROM shop.product_type;
```

**Result Grid:**

id	name
1	Платье
2	Джинсы
3	Шляпа
NONE	NONE

№	Действие	Команда на языке SQL и скрин результата
1	Создать таблицу Products	 <pre> CREATE TABLE `shop`.`product` (   `id` INT NOT NULL,   `brand_id` INT NOT NULL,   `product_type_id` INT NOT NULL,   `category_id` INT NOT NULL,   `price` DECIMAL(10,2) NOT NULL,   PRIMARY KEY (`id`) );  SELECT * FROM shop.product; </pre>

2

Заполнить таблицу  
Products

The screenshot shows the MySQL Workbench interface. In the top tab bar, there are tabs for 'product', 'Query 1', 'products', 'category', and another 'products' tab. The 'Query 1' tab contains the following SQL code:

```
1 • SELECT * FROM shop.products;
2 • use shop;
3
4 • INSERT INTO products (id, brand_id, product_type_id, category_id, price) VALUES (2, 2, 1, 1, 1999);
5 • INSERT INTO products (id, brand_id, product_type_id, category_id, price) VALUES (3, 2, 2, 1, 3999);
6 • INSERT INTO products (id, brand_id, product_type_id, category_id, price) VALUES (4, 2, 2, 2, 4999);
```

The 'Result Grid' tab is selected, showing the following data:

id	brand_id	product_type_id	category_id	price
1	1	1	1	1999.00
2	2	1	1	1999.00
3	2	2	1	3999.00
4	2	2	2	4999.00
NULL	NULL	NULL	NULL	NULL

use shop;

INSERT INTO products (id, brand\_id, product\_type\_id, category\_id, price) VALUES (2, 2, 1, 1, 1999);

INSERT INTO products (id, brand\_id, product\_type\_id, category\_id, price) VALUES (3, 2, 2, 1, 3999);

INSERT INTO products (id, brand\_id, product\_type\_id, category\_id, price) VALUES (4, 2, 2, 2, 4999);

10 lesson

## УРОК 10. Настройка согласованности данных ВНЕШНИЙ КЛЮЧ - FOREIGN KEY constraints

The screenshot shows the MySQL Workbench interface with two queries and their results.

**Query 1:** products

```
1 • SELECT * FROM shop.products;
```

**Result Grid:**

id	brand_id	product_type_id	category_id	price
1	1	1	1	1999.00
2	2	1	1	1999.00
3	2	2	1	3999.00
4	2	2	2	4999.00
NULL	NULL	NULL	NULL	NULL

**Query 2:** products

```
1 • SELECT * FROM shop.brand;
```

**Result Grid:**

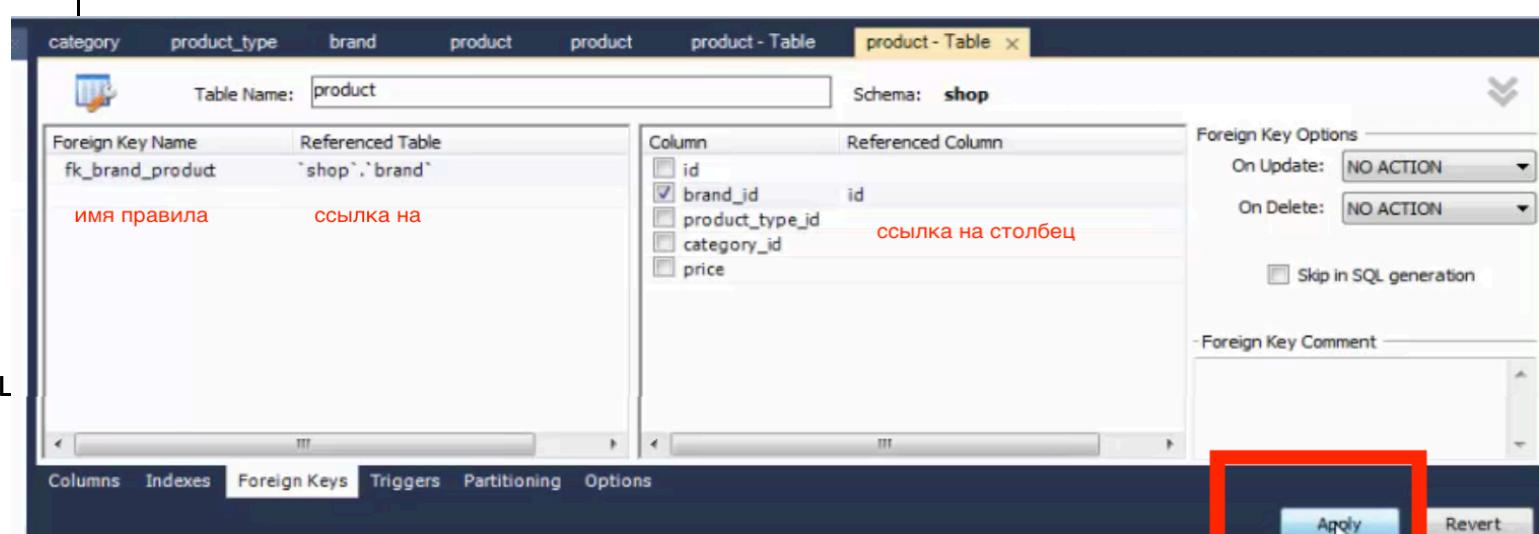
id	name
1	Marc O'Polo
2	Alcott
3	GUESS
4	ADIDAS
5	H&M
NULL	NULL

A red box highlights the **brand\_id** column in the products table result grid, and another red box highlights the **id** column in the brand table result grid. A red arrow points from the text "Настройка этой связи консистентности" to the brand\_id column in the products table result grid.

Настройка этой связи  
консистентности

1

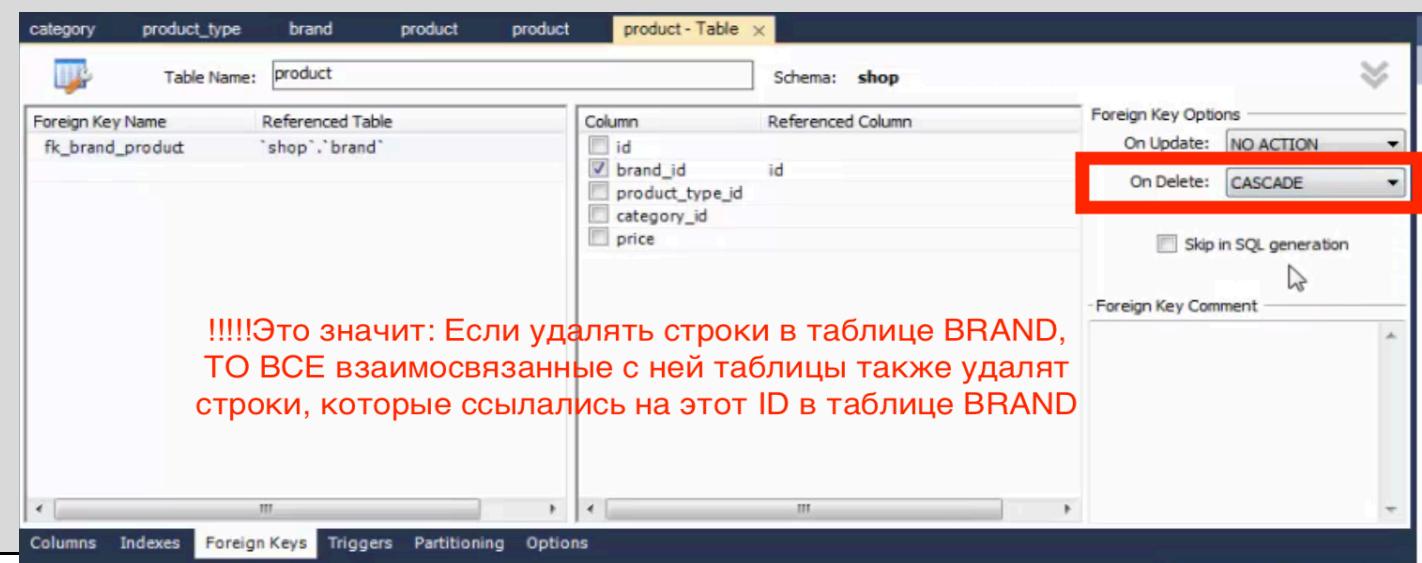
Настройка связи  
между столбцом  
brand\_id в таб.  
Products со столбцом  
id в таб. Brand



shop - products (прав.кл. мыши) - Alter Table - вкладка внизу Foreign  
Keys -

2

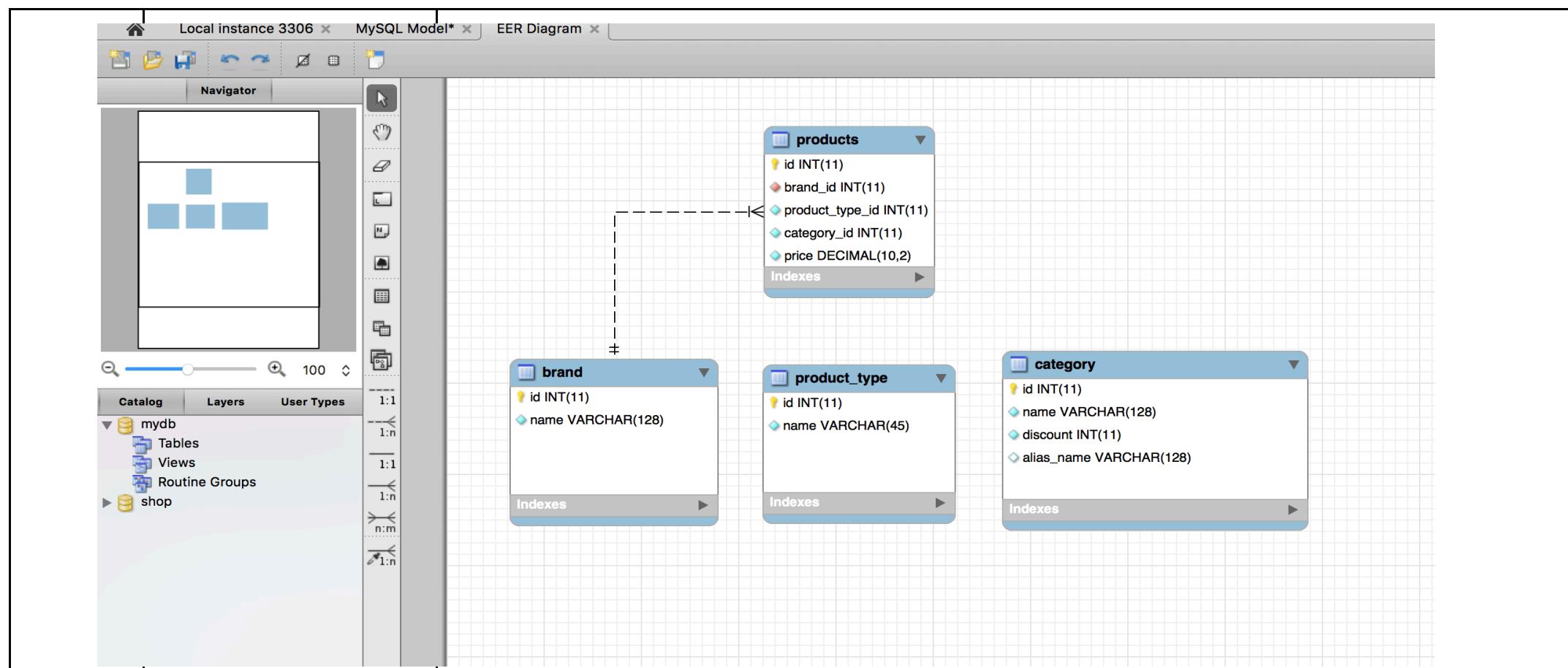
!!!!!!Настройка  
ОПЦИОНАЛЬНОСТИ



3

Визуализация  
связей

Основное меню программы - Database - Reverse Engineer - (выбираем нашу базу данных)  
будет такой результат. Где видны связи между таблицами



4

Настройка связей  
между таб. Products  
с таблицами type,  
category

```
ALTER TABLE `shop`.`products`  
ADD INDEX `fk_type_product_idx` (`product_type_id` ASC),  
ADD INDEX `fk_category_product_idx` (`category_id` ASC);  
ALTER TABLE `shop`.`products`  
ADD CONSTRAINT `fk_type_product`  
FOREIGN KEY (`product_type_id`)  
REFERENCES `shop`.`product_type` (`id`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION,  
ADD CONSTRAINT `fk_category_product`  
FOREIGN KEY (`category_id`)  
REFERENCES `shop`.`category` (`id`)  
ON DELETE NO ACTION  
ON UPDATE NO ACTION;
```

product    Query 1    products    category    products    brand    products

**Name:** products

Foreign Key	Referenced Table
fk_brand_product	`shop`.`brand`
fk_type_product	`shop`.`product_type`
fk_category_product	`shop`.`category`
<click to edit>	

Foreign key details 'fk\_category\_product'

Column	Referenced Column
<input type="checkbox"/> id	
<input type="checkbox"/> brand_id	
<input type="checkbox"/> product_type_id	
<input checked="" type="checkbox"/> category_id	id
<input type="checkbox"/> price	

Apply SQL Script to Database

### Review the SQL Script to be Applied on the Database

Please review the following SQL script that will be applied to the database.  
 Note that once applied, these statements may not be revertible without losing some  
 You can also manually change the SQL statements before execution.

Online DDL

Algorithm: Default    Lock Type: Default

```

1  ALTER TABLE `shop`.`products`
2   ADD INDEX `fk_type_product_idx` (`product_type_id` ASC),
3   ADD INDEX `fk_category_product_idx` (`category_id` ASC);
4  ALTER TABLE `shop`.`products`
5   ADD CONSTRAINT `fk_type_product`
6     FOREIGN KEY (`product_type_id`)
7       REFERENCES `shop`.`product_type` (`id`)
8       ON DELETE NO ACTION
9       ON UPDATE NO ACTION,
10  ADD CONSTRAINT `fk_category_product`
11    FOREIGN KEY (`category_id`)
12      REFERENCES `shop`.`category` (`id`)
13      ON DELETE NO ACTION
14      ON UPDATE NO ACTION;
15

```

11 lesson		<b>Урок 11. Создание таблиц с отношением “многие ко многим”. Создаем таблицу ORDER для хранения заказа в интернет магазине, ORDER_PRODUCTS - связывающая таблица заказов из ORDER с товарами из PRODUCTS</b>
1	<p>Создаем <u>таблицу</u>  <b>ЗАКАЗОВ ORDER</b> в          которой будет          храниться          информация о          заказе</p>	<pre>CREATE TABLE `shop`.`order` ( `id` INT NOT NULL AUTO_INCREMENT, `user_name` VARCHAR(128) NOT NULL, `user_phone` VARCHAR(45) NOT NULL, `date_time_order` DATETIME NOT NULL, PRIMARY KEY (`id`));</pre>

order - Table × product × Query 1 × products × category × products × brand

Name: order

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
id	INT	✓	✓					✓		
user_name	VARCHAR(128)		✓							
user_phone	VARCHAR(45)		✓							
date_time_order	DATETIME		✓							
<click to edit>										

You can also manually change the SQL statements before execution.

Algorithm: Default Lock Type: Default

```
CREATE TABLE `shop`.`order` (
  `id` INT NOT NULL AUTO_INCREMENT,
  `user_name` VARCHAR(128) NOT NULL,
  `user_phone` VARCHAR(45) NOT NULL,
  `date_time_order` DATETIME NOT NULL,
  PRIMARY KEY (`id`));

```

order - Table × product × Query 1 ×

1 ● SELECT \* FROM shop.order;

Result Grid | Filter Rows: Search

id	user_name	user_phone	date_time_order
NULL	NULL	NULL	NULL

Columns Indexes Foreign Keys Triggers Partitioning Options

The screenshot shows the MySQL Workbench interface. On the left, the 'order - Table' tab is selected, displaying the table structure with four columns: id (INT, PK, AI), user\_name (VARCHAR(128)), user\_phone (VARCHAR(45)), and date\_time\_order (DATETIME). The 'Default / Expression' column shows checkboxes for various constraints. Below the table definition, a SQL editor window contains the CREATE TABLE statement for the 'order' table. On the right, the 'Query 1' tab is active, showing the result of the SELECT \* FROM shop.order query, which returns four rows of NULL values for all columns. Both the table creation and the query results are highlighted with red boxes.

2

Заполняем ORDER. В нем храним данные по user и НЕ храним данные по выбранному товару

The screenshot shows the MySQL Workbench interface with the 'order' table selected. The SQL editor contains the following code:

```
1 • SELECT * FROM shop.`order`;
2 • INSERT INTO shop.`order` (user_name, user_phone, date_time_order) VALUES ('Vasya', '555-55-66', '2019-05-05 14:20');
```

The result grid shows the following data:

id	user_name	user_phone	date_time_order
1	Vasya	555-55-66	2019-05-05 14:20:00
	NULL	NULL	NULL

**INSERT INTO shop.`order` (user\_name, user\_phone, date\_time\_order) VALUES ('Vasya', '555-55-66', '2019-05-05 14:20');**

3

Создаем таблицу взаимосвязи заказов и товаров ORDER PRODUCTS в которой будет храниться информация о выбранном в заказе товаре user-ом

The screenshot shows the MySQL Workbench interface with the 'order\_products' table selected. The table has three columns: 'order\_id' (PK, INT(11), NOT NULL), 'product\_id' (INT(11), NULL), and 'count' (INT(11), NULL). The primary key is set to 'order\_id'. The 'Result Grid' shows a single row with all columns set to NULL.

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
order_id	INT(11)	✓	✓	✓						id из ORDER
product_id	INT(11)	✓	✓	✓						id из PRODUCT
count	INT(11)		✓							количество

```
CREATE TABLE `shop`.`order_products` (
  `order_id` INT(11) NOT NULL,
  `product_id` INT(11) NULL,
  `count` INT(11) NULL,
  PRIMARY KEY (`order_id`));
```

**CREATE TABLE`shop`(`order\_products` (**  
**`order\_id` INT(11) NOT NULL,**  
**`product\_id` INT(11) NULL,**  
**`count` INT(11) NULL,**  
**PRIMARY KEY (order\_id));**

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
order_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	id из ORDER					
product_id	INT(11)	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	id из PRODUCT					
count	INT(11)	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	количество
<click to edit>		<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	

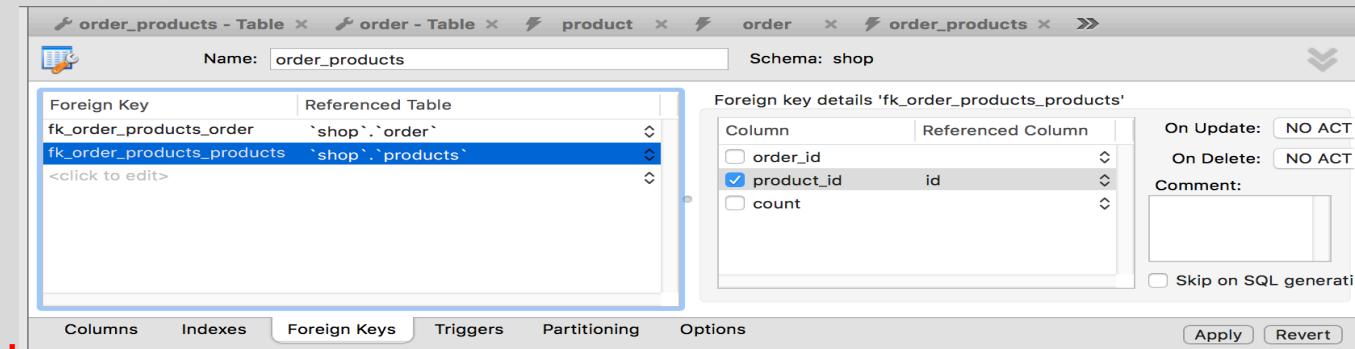
Составной первичный ключ, состоит из двух и более столбцов. Это значит что в таблице не может быть две строки с одинаковым ORDER\_ID и PRODUCT\_ID

12 lesson

## Урок 12. Составной первичный ключ. Создаем таблицу ORDER\_PRODUCTS - связывающая таблица заказов из ORDER с товарами из PRODUCTS

4

Настройка связей в таблице ORDER\_PRODUCTS устанавливаем ключи и связываем ее с PRODUCTS и ORDER (настройка внешних ключей foreign keys)



```
ALTER TABLE `shop`.`order_products`
ADD INDEX `fk_order_products_products_idx` (`product_id` ASC);
ALTER TABLE `shop`.`order_products`
ADD CONSTRAINT `fk_order_products_order`
FOREIGN KEY (`order_id`)
REFERENCES `shop`.`order` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION,
ADD CONSTRAINT `fk_order_products_products`
FOREIGN KEY (`product_id`)
REFERENCES `shop`.`products` (`id`)
ON DELETE NO ACTION
ON UPDATE NO ACTION;
```

5

Визуализируем связи	<p><b>Основное меню программы - Database - Reverse Engineer - (выбираем нашу базу данных)</b></p> <p><b>будет такой результат. Где видны связи между таблицами</b></p> <pre>     graph TD         product_type[product_type] &lt;--&gt; &lt;--&gt;  products[products]         products &lt;--&gt; &lt;--&gt;  brand[brand]         category[category] &lt;--&gt; &lt;--&gt;  products         order_products[order_products] --&gt; &gt;--&gt;  order[order]     </pre>
------------------------	--

## My Database

The screenshot shows a database management interface with six tables displayed in separate windows:

- products**: Shows a grid of data with columns: id, brand\_id, product\_type\_id, category\_id, and price. The data includes rows for various products like платье, джинсы, шляпа, and шуба.
- product\_type**: Shows a grid of data with columns: id and name. The data includes rows for Платье, Джинсы, Шляпа, and Шуба.
- brand**: Shows a grid of data with columns: id and name. The data includes rows for Marc O'Polo, Alcott, GUESS, ADIDAS, H&M, and others.
- order\_products**: Shows a grid of data with columns: order\_id, product\_id, and count. The data includes rows for multiple occurrences of each product ID.
- category**: Shows a grid of data with columns: id, name, discount, and alias\_name. The data includes rows for Женская одежда, Мужская одежда, Детская одежда, Головные уборы, Летние головные уборы, Женское нижнее белье, Мужское нижнее белье, and Мужская обувь.
- order**: Shows a grid of data with columns: id, user\_name, user\_phone, and date\_time\_order. The data includes rows for Vasya, Petya, Sasha, Nina, and Kate.

13 lesson

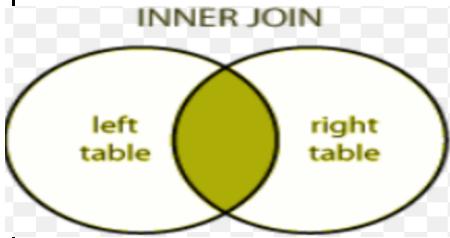
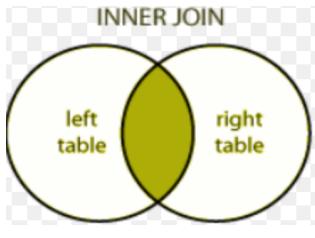
## Урок 13. Объединение данных из нескольких таблиц (INNER JOIN, LEFT JOIN \ RIGHT JOIN, FULL OUTER JOIN)

## в лекции немного теории\логики на примере xls

## My Database

The screenshot shows a database management interface with six tables displayed in separate windows:

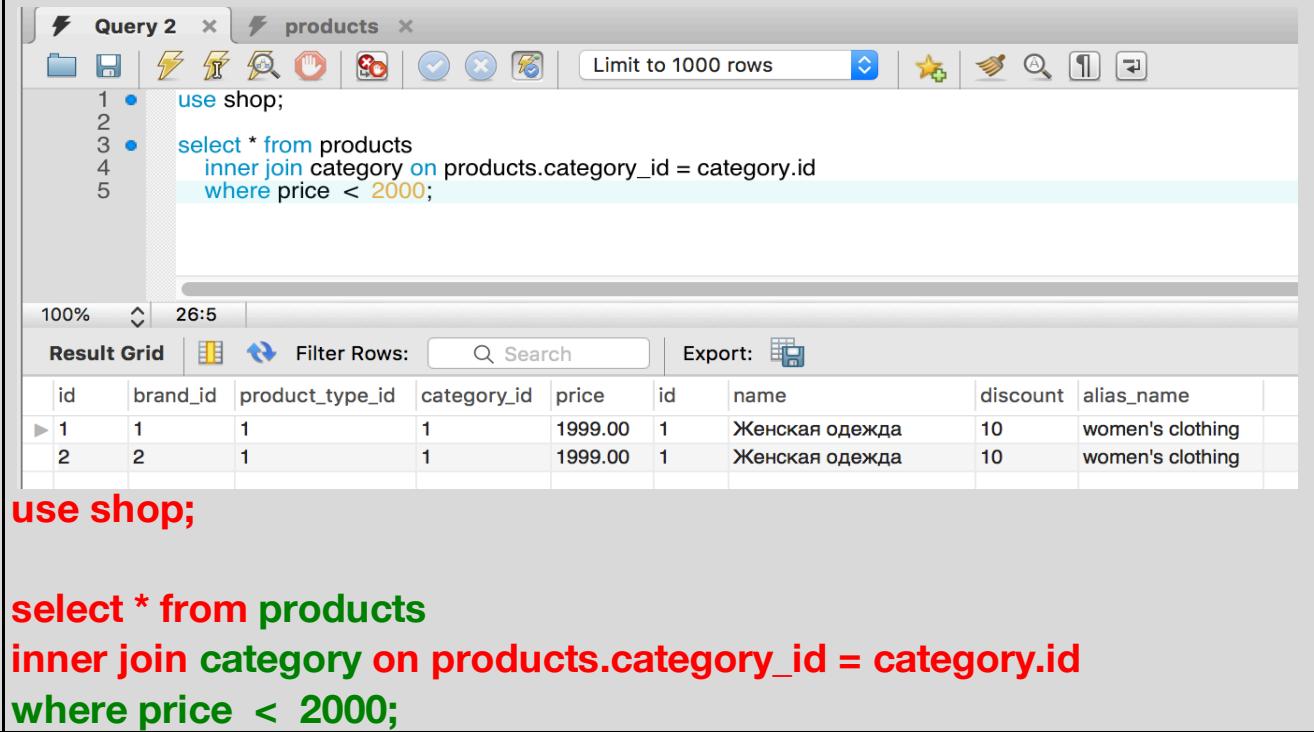
- products**: Shows a grid of products with columns: id, brand\_id, product\_type\_id, category\_id, and price. Data includes rows for various products like платье, джинсы, шапка, etc., with prices ranging from 33.00 to 1999.00.
- product\_type**: Shows a grid of product types with columns: id and name. Data includes Платье, Джинсы, Шапка, etc.
- brand**: Shows a grid of brands with columns: id and name. Data includes Marc O'Polo, Alcott, GUESS, ADIDAS, H&M, etc.
- order\_products**: Shows a grid of order products with columns: order\_id, product\_id, and count. Data includes multiple entries for each order\_id.
- category**: Shows a grid of categories with columns: id, name, discount, and alias\_name. Data includes Женская одежда, Мужская одежда, Детская одежда, Головные уборы, Летние головные уборы, Женское нижнее белье, Мужское нижнее белье, and Мужская обувь.
- order**: Shows a grid of orders with columns: id, user\_name, user\_phone, and date\_time\_order. Data includes orders placed by Vasya, Petya, Sasha, Nina, and Kate at various dates and times.

14 lesson		<b>Урок 14. Объединение данных из нескольких таблиц (INNER JOIN)</b>																																																	
№	<b>Действие</b>	<b>Команда на языке SQL и скрин результата</b>																																																	
		<p><b>ОПЕРАТОР INNER JOIN -Отображает то, что является пересечением двух таблиц по условию описанному в ON</b></p>																																																	
1	<p>Объединение данных в таблице PRODUCTS с данными из CATEGORY. Знак * - означает вывести ВСЕ ДАННЫЕ</p> 	<p>Хочу таб. products объединить с таб. category</p> <p>По принципу category_id в таб. products равен id в таб. category</p> <pre> use shop; select * from products inner join category on products.category_id = category.id;</pre> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">Result Grid</th> <th style="text-align: right;">Filter Rows:</th> <th style="text-align: right;">Export:</th> </tr> </thead> <tbody> <tr> <td style="vertical-align: top;"> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">id</th> <th style="text-align: left;">brand_id</th> <th style="text-align: left;">product_type_id</th> <th style="text-align: left;">category_id</th> <th style="text-align: left;">price</th> <th style="text-align: left;">id</th> <th style="text-align: left;">name</th> <th style="text-align: left;">discount</th> <th style="text-align: left;">alias_name</th> </tr> </thead> <tbody> <tr> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">3</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">1</td> <td style="text-align: left;">3999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">4</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">4999.00</td> <td style="text-align: left;">2</td> <td style="text-align: left;">Мужская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">man's clothing</td> </tr> </tbody> </table> </td> </tr> </tbody></table> <p style="color: red;"> use shop;  select * from products  inner join category on products.category_id = category.id; </p>	Result Grid	Filter Rows:	Export:	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">id</th> <th style="text-align: left;">brand_id</th> <th style="text-align: left;">product_type_id</th> <th style="text-align: left;">category_id</th> <th style="text-align: left;">price</th> <th style="text-align: left;">id</th> <th style="text-align: left;">name</th> <th style="text-align: left;">discount</th> <th style="text-align: left;">alias_name</th> </tr> </thead> <tbody> <tr> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">3</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">1</td> <td style="text-align: left;">3999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">4</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">4999.00</td> <td style="text-align: left;">2</td> <td style="text-align: left;">Мужская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">man's clothing</td> </tr> </tbody> </table>	id	brand_id	product_type_id	category_id	price	id	name	discount	alias_name	1	1	1	1	1999.00	1	Женская одежда	10	women's clothing	2	2	1	1	1999.00	1	Женская одежда	10	women's clothing	3	2	2	1	3999.00	1	Женская одежда	10	women's clothing	4	2	2	2	4999.00	2	Мужская одежда	10	man's clothing
Result Grid	Filter Rows:	Export:																																																	
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: left;">id</th> <th style="text-align: left;">brand_id</th> <th style="text-align: left;">product_type_id</th> <th style="text-align: left;">category_id</th> <th style="text-align: left;">price</th> <th style="text-align: left;">id</th> <th style="text-align: left;">name</th> <th style="text-align: left;">discount</th> <th style="text-align: left;">alias_name</th> </tr> </thead> <tbody> <tr> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1</td> <td style="text-align: left;">1999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">3</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">1</td> <td style="text-align: left;">3999.00</td> <td style="text-align: left;">1</td> <td style="text-align: left;">Женская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">women's clothing</td> </tr> <tr> <td style="text-align: left;">4</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">2</td> <td style="text-align: left;">4999.00</td> <td style="text-align: left;">2</td> <td style="text-align: left;">Мужская одежда</td> <td style="text-align: left;">10</td> <td style="text-align: left;">man's clothing</td> </tr> </tbody> </table>	id	brand_id	product_type_id	category_id	price	id	name	discount	alias_name	1	1	1	1	1999.00	1	Женская одежда	10	women's clothing	2	2	1	1	1999.00	1	Женская одежда	10	women's clothing	3	2	2	1	3999.00	1	Женская одежда	10	women's clothing	4	2	2	2	4999.00	2	Мужская одежда	10	man's clothing						
id	brand_id	product_type_id	category_id	price	id	name	discount	alias_name																																											
1	1	1	1	1999.00	1	Женская одежда	10	women's clothing																																											
2	2	1	1	1999.00	1	Женская одежда	10	women's clothing																																											
3	2	2	1	3999.00	1	Женская одежда	10	women's clothing																																											
4	2	2	2	4999.00	2	Мужская одежда	10	man's clothing																																											

4

Объединение данных в таблице PRODUCTS с данными из CATEGORY.

Знак \* - означает вывести ВСЕ ДАННЫЕ  
НО!!!!!! Только те, где  
price < 2000



The screenshot shows a MySQL Workbench interface. In the top-left pane, there are two tabs: "Query 2" and "products". The "products" tab is active. Below the tabs is a toolbar with various icons. The main area contains a code editor with the following SQL query:

```
use shop;
select * from products
inner join category on products.category_id = category.id
where price < 2000;
```

Below the code editor is a status bar showing "100%" and "26:5". The bottom half of the window is a "Result Grid" displaying the query results. The grid has the following columns: id, brand\_id, product\_type\_id, category\_id, price, id, name, discount, and alias\_name. There are two rows of data:| id | brand\_id | product\_type\_id | category\_id | price | id | name | discount | alias\_name |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 1 | 1 | 1 | 1999.00 | 1 | Женская одежда | 10 | women's clothing |
| 2 | 2 | 1 | 1 | 1999.00 | 1 | Женская одежда | 10 | women's clothing |

Below the result grid, the query is repeated in red text:

**use shop;**

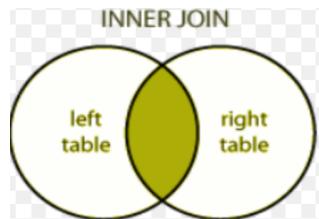
**select \* from products**

**inner join category on products.category\_id = category.id**

**where price < 2000;**

2

Объединение данных в  
таблице PRODUCTS с  
данными из CATEGORY.  
ПО ВЫБОРОЧНЫМ  
ПОЛЯМ



Query 2 products

```
1 • use shop;      Хочу таб. products объединить с таб. category
2
3
4 • select products.id, price, name from products
   inner join category on products.category_id = category.id;
5
```

По принципу category\_id в таб. products равен id в таб. category

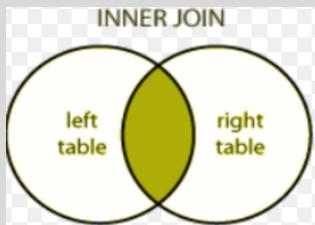
Но берем только id и price из products, и name из category

Result Grid

id	price	name
1	1999.00	Женская одежда
2	1999.00	Женская одежда
3	3999.00	Женская одежда
4	4999.00	Мужская одежда

use shop;  
select products.id, price, name from products  
inner join category on products.category\_id = category.id;

3



Объединение данных в таблице CATEGORY с данными из PRODUCTS.  
Знак \* - означает вывести ВСЕ ДАННЫЕ

Query 2 x products x

use shop; Хочу таб. category объединить с таб. products

select \* from category  
inner join products on products.category\_id = category.id;

По принципу category\_id в таб. products равен id в таб. category

Данные из category				Данные из products				
id	name	discount	alias_name	id	brand_id	product_type_id	category_id	price
1	Женская одежда	10	women's clothing	1	1	1	1	1999.00
1	Женская одежда	10	women's clothing	2	2	1	1	1999.00
1	Женская одежда	10	women's clothing	3	2	2	1	3999.00
2	Мужская одежда	10	man's clothing	4	2	2	2	4999.00

use shop;  
select \* from category  
inner join products on products.category\_id = category.id;

5

Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT\_TYPE.  
Знак \* - означает вывести ВСЕ ДАННЫЕ

The screenshot shows a MySQL Workbench interface. The query editor at the top contains the following SQL code:

```
use shop;
select * from products
inner join category on products.category_id = category.id
inner join brand on brand.id = products.brand_id
inner join product_type on product_type.id = products.product_type_id;
```

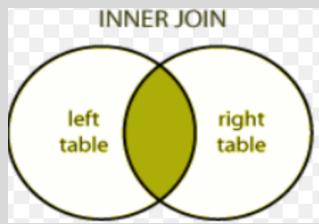
The result grid below displays the joined data from four tables. The columns are labeled: id, brand\_id, product\_type\_id, category\_id, price, id, name, discount, alias\_name, id, name, id, name. The data shows four rows of products, each associated with a category, brand, and product type.

id	brand_id	product_type_id	category_id	price	id	name	discount	alias_name	id	name	id	name
1	1	1	1	1999.00	1	Женская одежда	10	women's clothing	1	Marc O'Polo	1	Платье
2	2	1	1	1999.00	1	Женская одежда	10	women's clothing	2	Alcott	1	Платье
3	2	2	1	3999.00	1	Женская одежда	10	women's clothing	2	Alcott	2	Джинсы
4	2	2	2	4999.00	2	Мужская одежда	10	man's clothing	2	Alcott	2	Джинсы

Below the grid, the query is displayed again in red text:

```
use shop;  
  
select * from products  
inner join category on products.category_id = category.id  
inner join brand on brand.id = products.brand_id  
inner join product_type on product_type.id = products.product_type_id;
```

6



Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT\_TYPE.  
ВЫБОРОЧНЫЕ СТОЛБЦЫ

Query 2    products    brand    product\_type

use shop;

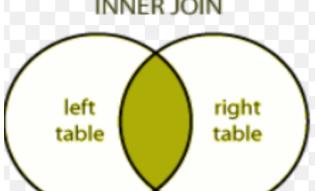
select products.id, brand.name, product\_type.name, category.name, products.price from products  
inner join category on products.category\_id = category.id  
inner join brand on brand.id = products.brand\_id  
inner join product\_type on product\_type.id = products.product\_type\_id;

Result Grid | Filter Rows: | Search | Export:

id	name	name	name	price
1	Marc OPolo	Платье	Женская одежда	1999.00
2	Alcott	Платье	Женская одежда	1999.00
3	Alcott	Джинсы	Женская одежда	3999.00
4	Alcott	Джинсы	Мужская одежда	4999.00

use shop;  
select products.id, brand.name, product\_type.name, category.name,  
products.price from products  
inner join category on products.category\_id = category.id  
inner join brand on brand.id = products.brand\_id  
inner join product\_type on product\_type.id = products.product\_type\_id;

7 ДЗ

INNER JOIN  
  
Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT\_TYPE.  
Знак \* - означает вывести ВСЕ ДАННЫЕ,  
НО!!!! Только  
ДЖИНСЫ!!!!

Query 2 x products x brand x product\_type x

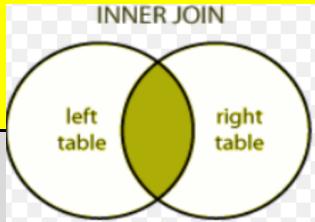
use shop;

select \* from products  
inner join category on products.category\_id = category.id  
inner join brand on brand.id = products.brand\_id  
inner join product\_type on product\_type.id = products.product\_type\_id  
where product\_type.id = 2;

Result Grid | Filter Rows: Search Export:

id	brand_id	product_type_id	category_id	price	id	name	discount	alias_name	id	name	id	name
3	2	2	1	3999.00	1	Женская одежда	10	women's clothing	2	Alcott	2	Джинсы
4	2	2	2	4999.00	2	Мужская одежда	10	man's clothing	2	Alcott	2	Джинсы

use shop;  
select \* from products  
inner join category on products.category\_id = category.id  
inner join brand on brand.id = products.brand\_id  
inner join product\_type on product\_type.id = products.product\_type\_id  
where product\_type.id = 2;

15 lesson		<b>Урок 15. Объединение данных из нескольких таблиц</b> (Операторы LEFT JOIN и RIGHT JOIN)
8 ДЗ с урока 14	<p>Объединение данных в таблице PRODUCTS с данными из CATEGORY, BRAND, PRODUCT_TYPE. ВЫБОРОЧНЫЕ СТОЛБЦЫ и !!! <u>Только джинсы (id 2 in product type)!!!!</u></p>	<pre>USE shop;  SELECT products.id, brand.name <u>as brand_name</u>, product_type.name <u>as product_type</u>, category.name <u>as category_name</u>, products.price <b>FROM</b> products <u>INNER JOIN</u> category <b>ON</b> products.category_id = category.id <u>INNER JOIN</u> brand <b>ON</b> brand.id = products.brand_id <u>INNER JOIN</u> product_type <b>ON</b> product_type.id = products.product_type_id <b>WHERE</b> product_type.id = 2;</pre>

Query 2 × products × brand × product\_type ×

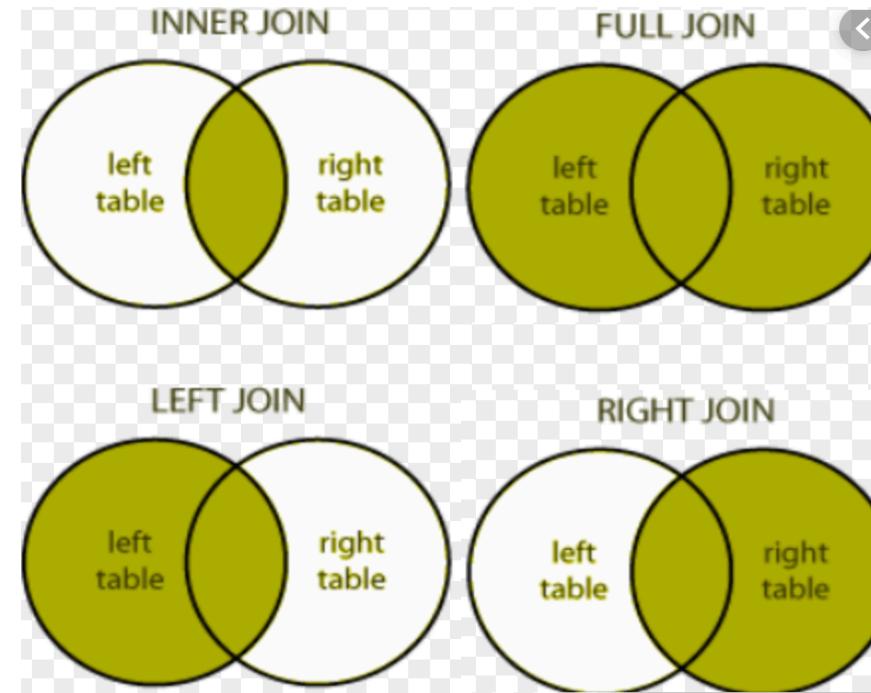
Limit to 1000 rows

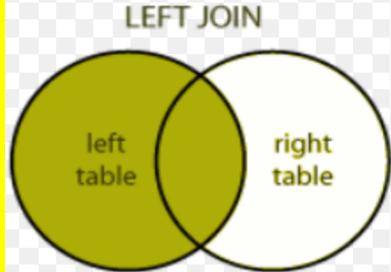
```
1 • USE shop;
2
3 • SELECT products.id, brand.name as brand_name, product_type.name as product_type, category.name as category_name, products.price FROM products
  INNER JOIN category ON products.category_id = category.id
  INNER JOIN brand ON brand.id = products.brand_id
  INNER JOIN product_type ON product_type.id = products.product_type_id
 WHERE product_type.id = 2;
```

100% 29:7

Result Grid | Filter Rows: Search | Export:

	id	brand_name	product_type	category_name	price	
3	3	Alcott	Джинсы	Женская одежда	3999.00	
4	4	Alcott	Джинсы	Мужская одежда	4999.00	

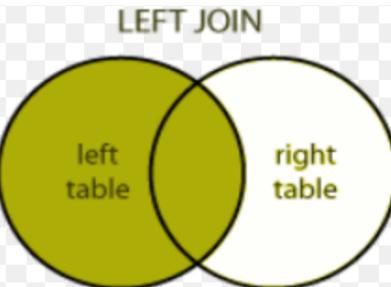




**ОПЕРАТОР LEFT JOIN - с его помощью можно посмотреть категории товаров, которых НЕТ в PRODUCTS.**

**Это объединение при котором в результирующий запрос попадает все из left table и строки из left table в которых есть пересечение с right table - при этом недостающие строчки заполняются NULL**

Покажи все из CATEGORY  
даже если этих  
продуктов НЕТ в  
PRODUCTS



```

Query 2 x products x brand x product_type x category x
File Edit View Insert Cell Tools Help
1
2 • SELECT * FROM category
3 LEFT JOIN products ON products.category_id = category.id;

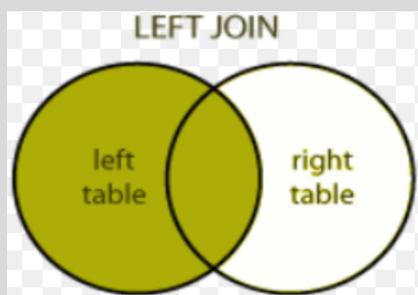
100% 59:3
Result Grid Filter Rows: Search Export:
id name discount alias_name id brand_id product_type_id category_id price
1 Женская одежда 10 women's clothing 1 1 1 1 1999.00
1 Женская одежда 10 women's clothing 2 2 1 1 1999.00
1 Женская одежда 10 women's clothing 3 2 2 1 3999.00
2 Мужская одежда 10 man's clothing 4 2 2 2 4999.00
3 Детская одежда 15 baby clothing NULL NULL NULL NULL NULL
4 Головные уборы 5 hats NULL NULL NULL NULL NULL
5 Летние головные уборы 3 summer hats NULL NULL NULL NULL NULL
6 Женское нижнее белье 7 women's secret clothing NULL NULL NULL NULL NULL
7 Мужское нижнее белье 6 men's secret clothing NULL NULL NULL NULL NULL
8 Мужская обувь 3 men's shoes NULL NULL NULL NULL NULL

```

**SELECT \* FROM category  
LEFT JOIN products ON products.category\_id = category.id;**

2

Если добавить к предыдущему запросу ограничение WHERE, то получим только те строки, по которым товар в products отсутствует



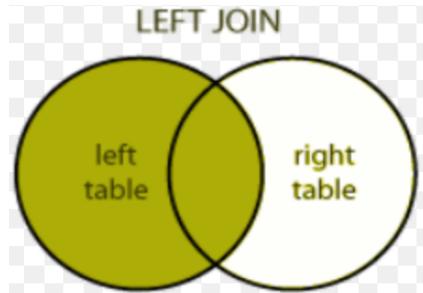
use shop;  
SELECT \* FROM category  
LEFT JOIN products ON products.category\_id = category.id  
WHERE products.id is null;

Result Grid | Filter Rows: Search | Export:

id	name	discount	alias_name	id	brand_id	product_type_id	category_id	price
3	Детская одежда	15	baby clothing	NULL	NULL	NULL	NULL	NULL
4	Головные уборы	5	hats	NULL	NULL	NULL	NULL	NULL
5	Летние головные уборы	3	summer hats	NULL	NULL	NULL	NULL	NULL
6	Женское нижнее белье	7	women's secret clothing	NULL	NULL	NULL	NULL	NULL
7	Мужское нижнее белье	6	men's secret clothing	NULL	NULL	NULL	NULL	NULL
8	Мужская обувь	3	men's shoes	NULL	NULL	NULL	NULL	NULL

3

Покажи товары из  
CATEGORY, КОТОРЫХ НЕТ  
В PRODUCTS. Столбцы с  
NULL НЕ ВЫВОДИТЬ



Query 2    products    brand    product\_type    category

use shop;  
SELECT category.\* FROM category  
LEFT JOIN products ON products.category\_id = category.id  
WHERE products.id is null;

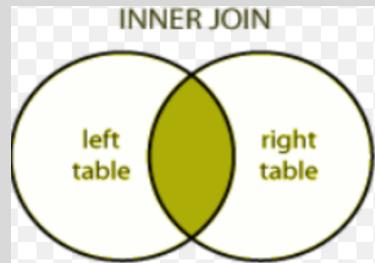
Result Grid    Filter Rows: Search    Export:

id	name	discount	alias_name
3	Детская одежда	15	baby clothing
4	Головные уборы	5	hats
5	Летние головные уборы	3	summer hats
6	Женское нижнее белье	7	women's secret clothing
7	Мужское нижнее белье	6	men's secret clothing
8	Мужская обувь	3	men's shoes

use shop;  
SELECT category.\* FROM category  
LEFT JOIN products ON products.category\_id = category.id  
WHERE products.id is null;

4 дз

Покажи все из ORDER\_PRODUCTS, что пересекается с PRODUCTS. Сделай по возрастанию столбец order\_id



```
use shop;  
select * from order_products  
inner join products on order_products.product_id = products.id  
ORDER BY order_id;
```

Screenshot of MySQL Workbench showing the execution of the query and the resulting data grid.

The query executed is:

```
use shop;  
select * from order_products  
inner join products on order_products.product_id = products.id  
ORDER BY order_id;
```

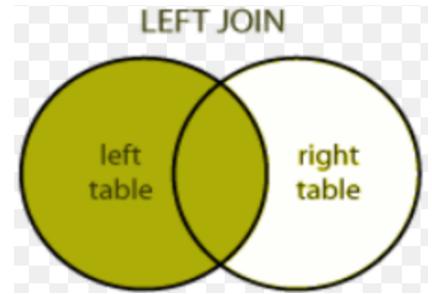
The Result Grid displays the following data:

order_id	product_id	count	id	brand_id	product_type_id	category_id	price
1	3	2	3	2	2	1	3999.00
1	1	1	1	1	1	1	1999.00
1	2	1	2	2	1	1	1999.00
2	3	1	3	2	2	1	3999.00
2	1	1	1	1	1	1	1999.00
2	2	2	2	2	1	1	1999.00
3	3	2	3	2	2	1	3999.00
3	1	2	1	1	1	1	1999.00
3	2	1	2	2	1	1	1999.00

The data is grouped into three orders (Zakaz 1, Zakaz 2, Zakaz 3) based on the order\_id column. A red arrow points to the order\_id column, and green boxes highlight the product\_id and id columns to show the join relationship. A blue box highlights the products table.

5 дз

Покажи все из  
PRODUCTS, даже если  
этих продуктов в заказах  
ORDER\_PRODUCTS НЕТ!



use shop;  
select \* from products  
left join order\_products on order\_products.product\_id = products.id;

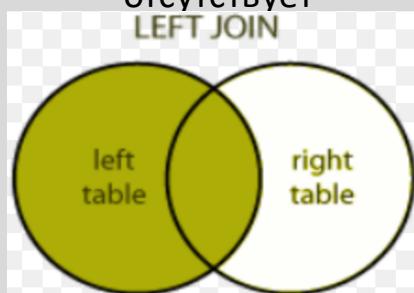
id	brand_id	product_type_id	category_id	price	order_id	product_id	count
1	1	1	1	1999.00	1	1	1
1	1	1	1	1999.00	2	1	1
1	1	1	1	1999.00	3	1	2
2	2	1	1	1999.00	1	2	1
2	2	1	1	1999.00	2	2	2
2	2	1	1	1999.00	3	2	1
3	2	2	1	3999.00	1	3	2
3	2	2	1	3999.00	2	3	1
3	2	2	1	3999.00	3	3	2
4	2	2	2	4999.00	NULL	NULL	NULL
5	2	1	2	33.00	NULL	NULL	NULL

use shop;  
select \* from products  
LEFT JOIN order\_products on order\_products.product\_id = products.id;

6дз

Если добавить к предыдущему запросу ограничение WHERE, то получим только те строки, по которым товар в ORDER\_PRODUCTS отсутствует

**LEFT JOIN**



Query 2    products    brand    product\_type    category  
Limit to 1000 rows

```
1 ● use shop;
2 ● select * from products
3     left join order_products on order_products.product_id = products.id
4     WHERE order_products.order_id is null;
```

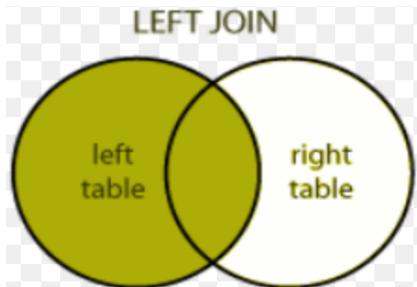
Result Grid    Filter Rows:    Search    Export:

id	brand_id	product_type_id	category_id	price	order_id	product_id	count
4	2	2	2	4999.00	NULL	NULL	NULL
5	2	1	2	33.00	NULL	NULL	NULL

use shop;  
select \* from products  
**LEFT JOIN order\_products on order\_products.product\_id = products.id**  
**WHERE order\_products.order\_id is null;**

**16 lesson****ДЗ с урока 15**1. ДЗ с  
урока 15

Покажи (объедини) все из PRODUCT\_TYPE даже то, чего нет в PRODUCTS



Query 2    category    order    order\_products    products

use shop;  
SELECT \* FROM product\_type  
LEFT JOIN products ON products.product\_type\_id = product\_type.id;

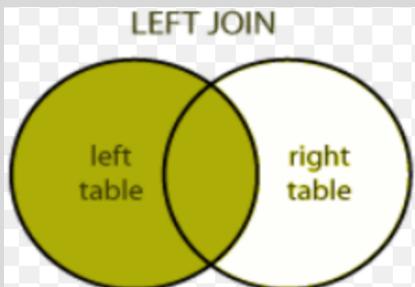
Result Grid    Filter Rows: Search    Export:

id	name	id	brand_id	product_type_id	category_id	price
1	Платье	1	1	1	1	1999.00
1	Платье	2	2	1	1	1999.00
2	Джинсы	3	2	2	1	3999.00
2	Джинсы	4	2	2	2	4999.00
1	Платье	5	2	1	2	33.00
3	Шляпа	NULL	NULL	NULL	NULL	NULL
4	шуба	NULL	NULL	NULL	NULL	NULL

use shop;  
SELECT \* FROM product\_type  
LEFT JOIN products ON products.product\_type\_id = product\_type.id;

2. ДЗ с  
урока 15

Покажи (объедини) все из PRODUCT\_TYPE даже то, чего нет в PRODUCTS, но выведи только то, что в условии WHERE



Query 2 × category × order × order\_products × products × category

use shop;

SELECT product\_type.\* FROM product\_type **Все только из product\_type**

LEFT JOIN products ON products.product\_type\_id = product\_type.id

WHERE product\_type\_id is null;

Result Grid | Filter Rows: Search | Export:

id	name
3	Шляпа
4	шуба

use shop;  
SELECT product\_type.\* FROM product\_type  
LEFT JOIN products ON products.product\_type\_id = product\_type.id  
WHERE product\_type\_id is null;

3. ДЗ

Вывести информацию  
обо всех товарах,  
которые НЕ ПОПАЛИ НИ  
В ОДИН ЗАКАЗ. Делаем  
это поэтапно. См. 3  
скрина

Query 14 x order\_products x order x

use shop;

SELECT \* FROM `order`  
INNER JOIN order\_products ON order\_products.order\_id = `order` .id;

Result Grid | Filter Rows: Search Export:

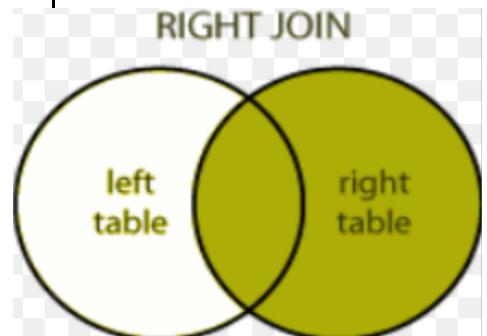
id	user_name	user_phone	date_time_order	order_id	product_id	count
1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1
1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1
1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2
2	Petya	111-55-66	2019-05-06 14:30:00	2	1	1
2	Petya	111-55-66	2019-05-06 14:30:00	2	2	2
2	Petya	111-55-66	2019-05-06 14:30:00	2	3	1
3	Sasha	222-22-22	2019-05-06 14:20:00	3	1	2
3	Sasha	222-22-22	2019-05-06 14:20:00	3	2	1
3	Sasha	222-22-22	2019-05-06 14:20:00	3	3	2

```
Query 14 x order_products x order x
use shop;
SELECT * FROM `order`
INNER JOIN order_products ON order_products.order_id = `order` .id
RIGHT JOIN products ON order_products.order_id = products.id;
```

100% 2:4

Result Grid Filter Rows: Search Export:

	id	user_name	user_phone	date_time_order	order_id	product_id	count	id	brand_id	product_type_id	category_id	price
>	1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1	1	1	1	1	1999.00
	1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1	1	1	1	1	1999.00
	1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2	1	1	1	1	1999.00
	2	Petya	111-55-66	2019-05-06 14:30:00	2	1	1	2	2	1	1	1999.00
	2	Petya	111-55-66	2019-05-06 14:30:00	2	2	2	2	2	1	1	1999.00
	2	Petya	111-55-66	2019-05-06 14:30:00	2	3	1	2	2	1	1	1999.00
	3	Sasha	222-22-22	2019-05-06 14:20:00	3	1	2	3	2	2	1	3999.00
	3	Sasha	222-22-22	2019-05-06 14:20:00	3	2	1	3	2	2	1	3999.00
	3	Sasha	222-22-22	2019-05-06 14:20:00	3	3	2	3	2	2	1	3999.00
	HULL	HULL	HULL	HULL	HULL	HULL	HULL	4	2	2	2	4999.00
	HULL	HULL	HULL	HULL	HULL	HULL	HULL	5	2	1	2	33.00



Query 14    order\_products    order

use shop;

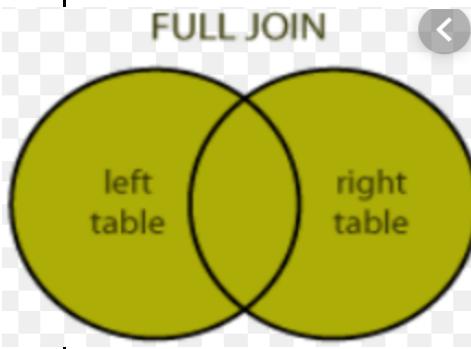
SELECT products.\* FROM `order`

INNER JOIN order\_products ON order\_products.order\_id = `order`.id  
RIGHT JOIN products ON order\_products.order\_id = products.id  
WHERE `order`.id IS NULL;

Result Grid | Filter Rows: Search Export:

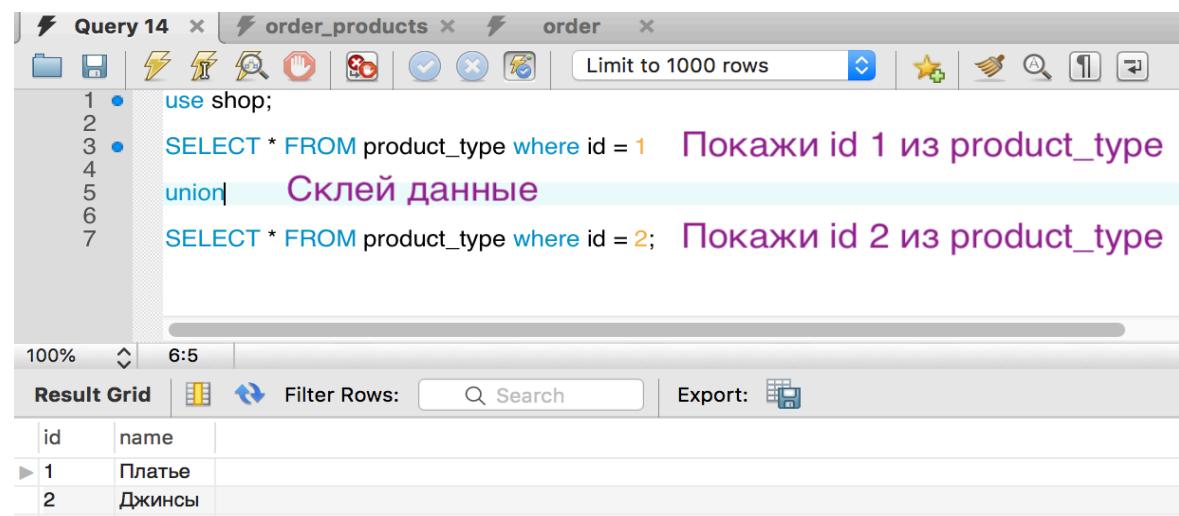
id	brand_...	product_type_id	category_id	price
4	2	2	2	4999.00
5	2	1	2	33.00

use shop;  
SELECT products.\* FROM `order`  
INNER JOIN order\_products ON order\_products.order\_id = `order`.id  
RIGHT JOIN products ON order\_products.order\_id = products.id  
WHERE `order`.id IS NULL;

16 lesson		ПОЛНОЕ объединение данных из НЕСКОЛЬКИХ таблиц (FULL JOIN FULL OUTER JOIN). UNION
		Полное объединение данных двух таблиц - ВСЕ без исключения строки обеих таблиц БЕЗ зависимости их пересечения. <b>В MySQL директивы FULL OUTER JOIN нет (она есть в MS SQL, Oracle, Post greSql).</b> В MySQL используют директиву UNION

1

принцип UNION -  
команда "СКЛЕЙ"  
строчки запросов,  
которые до и после  
**UNION**



The screenshot shows the MySQL Workbench interface with three tabs at the top: 'Query 14' (active), 'order\_products', and 'order'. The query editor contains the following code:

```
use shop;
SELECT * FROM product_type where id = 1; Покажи id 1 из product_type
union| Склей данные
SELECT * FROM product_type where id = 2; Покажи id 2 из product_type
```

The result grid shows two rows of data:

id	name
1	Платье
2	Джинсы

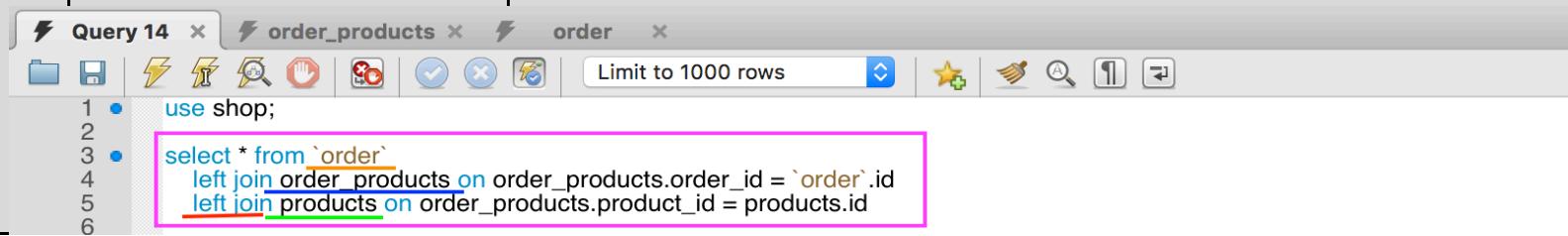
```
use shop;
SELECT * FROM product_type where id = 1
union
SELECT * FROM product_type where id = 2;
```

2

```
use shop;
select * from `order`
    left join order_products on order_products.order_id = `order`.id
    left join products on order_products.product_id = products.id

union

select * from `order`
    inner join order_products on order_products.order_id = `order`.id
    right join products on order_products.product_id = products.id
    where `order`.id is null;
```



The screenshot shows the MySQL Workbench interface with two queries in the editor. The first query, which includes the union operation and the second part of the union, is highlighted with a pink rectangle. The second query, which starts with 'select \* from `order`' and ends with 'where `order`.id is null;', is also visible below it.

```
Query 14 × order_products × order ×
1 ● use shop;
2 ● select * from `order`
3   left join order_products on order_products.order_id = `order`.id
4   left join products on order_products.product_id = products.id
5
6
Limit to 1000 rows
```

1 ● use shop;  
2 ● select \* from `order`  
3 left join order\_products on order\_products.order\_id = `order`.id  
4 left join products on order\_products.product\_id = products.id  
5  
6

```
7 union
8
9 select * from `order`
10 inner join order_products on order_products.order_id = `order`.id
11 right join products on order_products.product_id = products.id
12 where `order`.id is null;
13
```

100% ▾ 1:8

Result Grid



Filter Rows:

Search

Export:



	<b>id</b>	<b>user_name</b>	<b>user_phone</b>	<b>date_time_order</b>	<b>order_id</b>	<b>product_id</b>	<b>count</b>	<b>id</b>	<b>brand_id</b>	<b>product_type_id</b>	<b>category_id</b>	<b>price</b>
▶	1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1	1	1	1	1	1999.00
	2	Petya	111-55-66	2019-05-06 14:30:00	2	1	1	1	1	1	1	1999.00
	3	Sasha	222-22-22	2019-05-06 14:20:00	3	1	2	1	1	1	1	1999.00
	1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1	2	2	1	1	1999.00
	2	Petya	111-55-66	2019-05-06 14:30:00	2	2	2	2	2	1	1	1999.00
	3	Sasha	222-22-22	2019-05-06 14:20:00	3	2	1	2	2	1	1	1999.00
	1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2	3	2	2	1	3999.00
	2	Petya	111-55-66	2019-05-06 14:30:00	2	3	1	3	2	2	1	3999.00
	3	Sasha	222-22-22	2019-05-06 14:20:00	3	3	2	3	2	2	1	3999.00
	4	Nina	111-11-11	2019-05-08 14:20:00	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>
	5	Kate	333-33-33	2019-05-09 14:20:00	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>NUL</b>
					<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>4</b>	<b>2</b>	<b>2</b>	<b>2</b>	<b>4999.00</b>
					<b>NUL</b>	<b>NUL</b>	<b>NUL</b>	<b>5</b>	<b>2</b>	<b>1</b>	<b>2</b>	<b>33.00</b>

17

## Агрегирующие функции (count, sum, max, min)

№

Действие

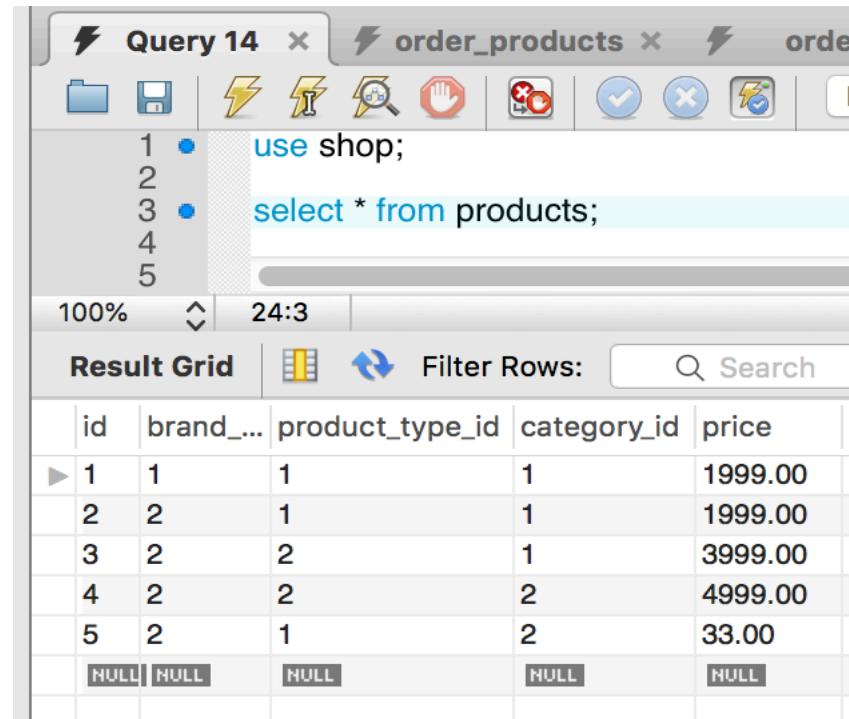
Команда на языке SQL и скрин результата

1

Покажи  
всю  
таблицу  
PRODUCTS

use shop;

select \* from products;



The screenshot shows a MySQL Workbench interface with three tabs at the top: 'Query 14', 'order\_products', and 'orders'. The 'Query 14' tab contains the following SQL code:

```
1 • use shop;
2
3 • select * from products;
```

The 'Result Grid' tab displays the following data from the 'products' table:

	id	brand_id	product_type_id	category_id	price
▶	1	1	1	1	1999.00
	2	2	1	1	1999.00
	3	2	2	1	3999.00
	4	2	2	2	4999.00
	5	2	1	2	33.00
	NULL	NULL	NULL	NULL	NULL

2

Покажи  
СКОЛЬКО  
строк в  
таблице  
PRODUCTS

**use shop;**

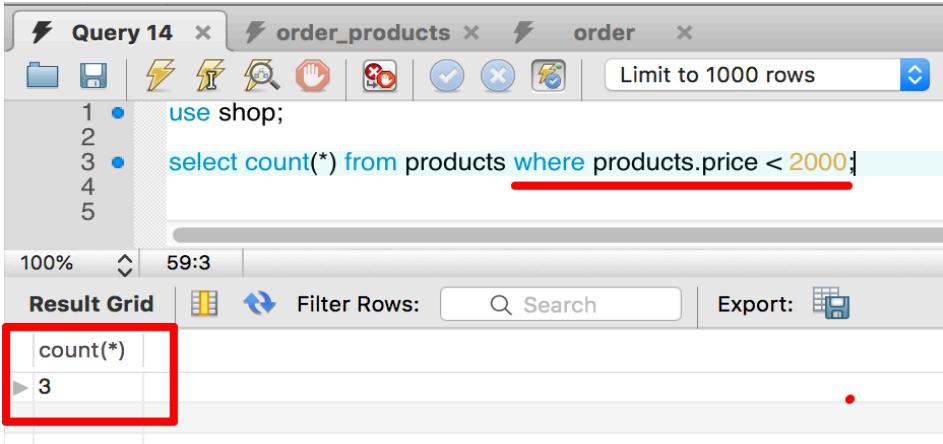
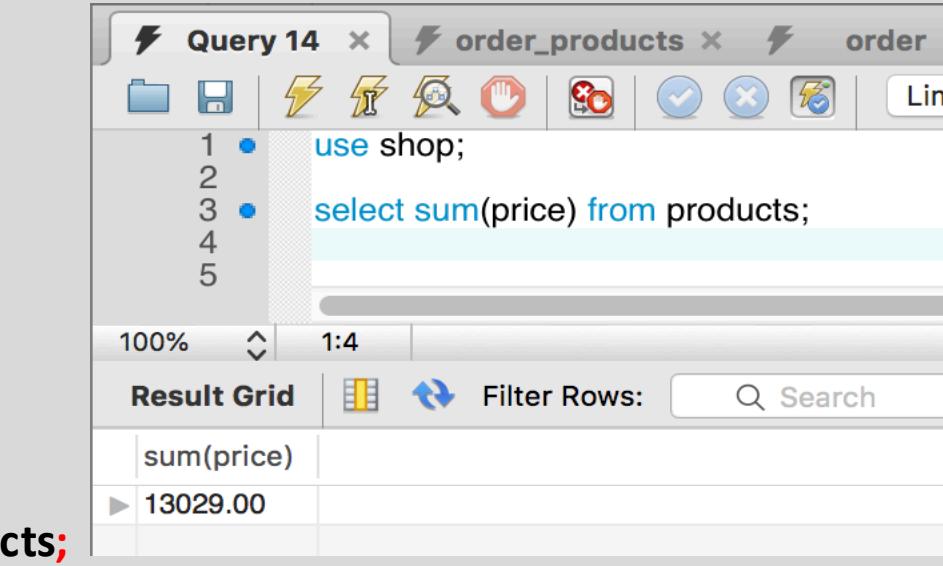
**select count(\*) from products;**

The screenshot shows a MySQL Workbench interface with two tabs: 'Query 14' and 'order\_products'. The 'Query 14' tab contains the following SQL code:

```
1 • use shop;
2
3 • select count(*) from products;
```

The 'Result Grid' shows the output of the query:

count(*)
5

3	<p>Покажи сколько строк в таблице PRODUCTS, удовлетворяющие условиям WHERE</p>	<p><b>use shop;</b></p> <p><b>select count(*) from products where products.price &lt; 2000;</b></p> 
4	<p>Покажи сумму столбца. Сумма столбца price в PRODUCTS</p>	<p><b>use shop;</b></p> <p><b>select sum(price) from products;</b></p> 

5

Покажи  
sum, min,  
max  
столбца  
price в  
PRODUCTS

**use shop;**

**select sum(price), min(price), max(price) from products;**

The screenshot shows a MySQL command-line interface. The query entered is:

```
1 • use shop;
2 • select sum(price), min(price), max(price) from products;
```

The results are displayed in a grid:

	sum(price)	min(price)	max(price)
▶	13029.00	33.00	4999.00

6

Покажи  
sum, min,  
max  
столбца  
price в  
PRODUCTS,  
с помощью  
as  
переимену  
й эти  
столбцы

The screenshot shows a MySQL Workbench interface. In the query editor, the following SQL code is written:

```
1 • use shop;
2
3 • select sum(price) as total_price, min(price) as min_price, max(price) as max_price from products;
```

The result grid shows the output of the query:

total_price	min_price	max_price
13029.00	33.00	4999.00

**use shop;**  
**select sum(price) as total\_price, min(price) as min\_price, max(price) as**  
**max\_price from products;**

7 дз

По ORDER  
и  
ORDER\_PRODUCTS  
видно, что  
товар есть  
только в  
корзинах  
Васи, Пети  
и Саши. У  
Нины и  
Кати  
товаров в

order    Query 14    order\_products

```
1 • use shop;
2
3 • SELECT * FROM shop.`order`;
4
5
```

Result Grid    Filter Rows: Search

id	user_name	user_phone	date_time_order
1	Vasya	555-55-66	2019-05-05 14:20:00
2	Petya	111-55-66	2019-05-06 14:30:00
3	Sasha	222-22-22	2019-05-06 14:20:00
4	Nina	111-11-11	2019-05-08 14:20:00
5	Kate	333-33-33	2019-05-09 14:20:00
NULL	NULL	NULL	NULL

В ORDER\_PRODUCTS видно, что товар лежит только в корзинах у Vasya, Petya, Sasha

order    order\_products    Query 14

```
1 • SELECT * FROM shop.order_products;
```

100% 1:1

Result Grid    Filter Rows: Search

order_id	product_id	count
1	1	1
1	2	1
1	3	2
2	1	1
2	2	2
2	3	1
3	1	2
3	2	1
3	3	2
NULL	NULL	NULL

Вывести  
суммарну  
ю  
стоимость  
заказа у

use shop;  
SELECT \*, price \* `count` as total\_price FROM `order`  
inner join order\_products on order\_products.order\_id = `order` . id  
inner join products on products.id = order\_products.product\_id  
where `order` . id = 1;

Vasya  
1 use shop;  
2  
3 SELECT \*, price \* `count` as total\_price FROM `order`  
4 inner join order\_products on order\_products.order\_id = `order` . id  
5 inner join products on products.id = order\_products.product\_id  
6  
7 where `order` . id = 1;  
8  
9  
10  
11

8 д

100% 1:10

Result Grid Filter Rows: Search Export:

	id	user_name	user_phone	date_time_order	order_id	product_id	count	id	brand_...	product_type_id	category_id	price	total_price
▶	1	Vasya	555-55-66	2019-05-05 14:20:00	1	1	1	1	1	1	1	1999.00	1999.00
	1	Vasya	555-55-66	2019-05-05 14:20:00	1	2	1	2	2	1	1	1999.00	1999.00
	1	Vasya	555-55-66	2019-05-05 14:20:00	1	3	2	3	2	2	1	3999.00	7998.00

9 дз

На основе  
данных из  
пункта  
выше  
вывести  
итого  
сумму по  
Васе.

Изменилас  
ь только в  
строка 3

```
1 • use shop;
2 • SELECT sum(price * `count`) as total_price FROM `order`
3   inner join order_products on order_products.order_id = `order` . id
4
5   inner join products on products.id = order_products.product_id
6
7   where `order` . id = 1;
```

Result Grid	
total_price	
11996.00	

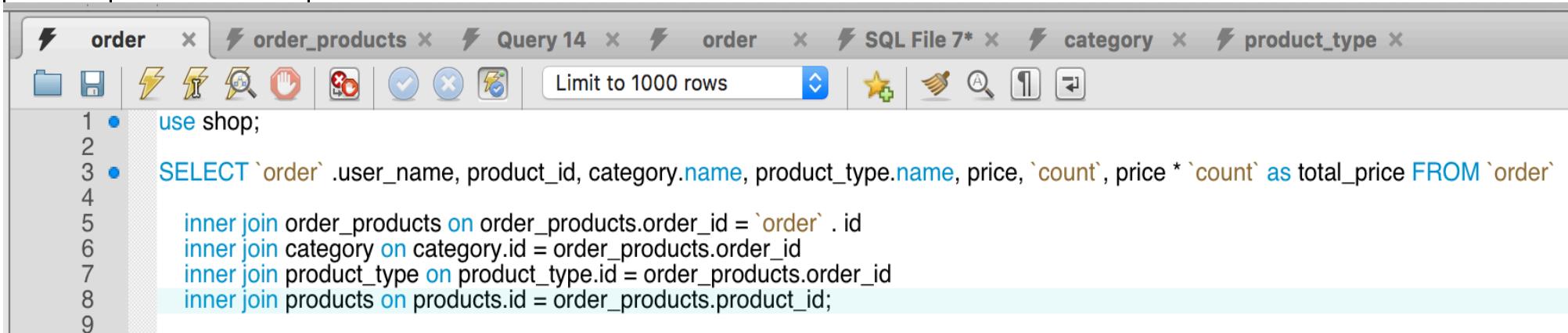
```
use shop;
SELECT sum(price * `count`) as total_price FROM `order`
inner join order_products on order_products.order_id = `order` . id
inner join products on products.id = order_products.product_id
where `order` . id = 1;
```

		<b>Оператор GROUP BY. Объединение по заказчику, вывод итого суммы по заказу, итого количество товаро в заказе (min, max, sum...)</b>
<b>№</b>	<b>Действие</b>	<b>Команда на языке SQL и скрин результата</b>
		Директива GROUP BY позволит группировать в одном запросе результаты агрегирующих функций для нескольких строк (получить суммарную стоимость заказа для каждого из пользователей)

1 Так  
выглядят  
заказы,  
видно, что  
у каждого  
покупателя  
отдельная  
строка на  
товар.

use shop;

```
SELECT `order` .user_name, product_id, category.name, product_type.name,
price, `count`, price * `count` as total_price FROM `order`  
  
inner join order_products on order_products.order_id = `order` . id  
inner join category on category.id = order_products.order_id  
inner join product_type on product_type.id = order_products.order_id  
inner join products on products.id = order_products.product_id;
```



The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** The main window displays the SQL query:use shop;  
SELECT `order` .user\_name, product\_id, category.name, product\_type.name, price, `count`, price \* `count` as total\_price FROM `order`  
inner join order\_products on order\_products.order\_id = `order` . id  
inner join category on category.id = order\_products.order\_id  
inner join product\_type on product\_type.id = order\_products.order\_id  
inner join products on products.id = order\_products.product\_id;
- Connections:** A tab bar at the top lists several connections: order, order\_products, Query 14, order, SQL File 7\*, category, and product\_type.
- Toolbar:** Below the tabs are various icons for file operations, search, and refresh.
- Code Area:** On the left, a vertical number column shows line numbers 1 through 9, with lines 1 and 2 highlighted in blue to indicate they are active.

10

100% 68:8

Result Grid Filter Rows: Search Export:

user_name	product_id	name	name	price	count	total_price
Vasya	1	Женская одежда	Платье	1999.00	1	1999.00
Vasya	2	Женская одежда	Платье	1999.00	1	1999.00
Vasya	3	Женская одежда	Платье	3999.00	2	7998.00
Petya	1	Мужская одежда	Джинсы	1999.00	1	1999.00
Petya	2	Мужская одежда	Джинсы	1999.00	2	3998.00
Petya	3	Мужская одежда	Джинсы	3999.00	1	3999.00
Sasha	1	Детская одежда	Шляпа	1999.00	2	3998.00
Sasha	2	Детская одежда	Шляпа	1999.00	1	1999.00
Sasha	3	Детская одежда	Шляпа	3999.00	2	7998.00

2

Или Так  
выглядят  
заказы,  
видно, что  
у каждого  
покупателя  
отдельная  
строка на  
товар.

```
use shop;
SELECT `order`.user_name, product_id, price, `count`, price * `count` as
total_price FROM `order`
inner join order_products on order_products.order_id = `order`. id
inner join products on products.id = order_products.product_id;
```

```
1 • use shop;
2
3 • SELECT `order` .user_name, product_id, price, `count`, price * `count` as total_price FROM `order`
4
5     inner join order_products on order_products.order_id = `order` . id
6
7     inner join products on products.id = order_products.product_id;
8
9
```

100% 1:9

Result Grid

Filter Rows:

Search

Export:

user_name	product_id	price	count	total_price
Vasya	1	1999.00	1	1999.00
Vasya	2	1999.00	1	1999.00
Vasya	3	3999.00	2	7998.00
Petya	1	1999.00	1	1999.00
Petya	2	1999.00	2	3998.00
Petya	3	3999.00	1	3999.00
Sasha	1	1999.00	2	3998.00
Sasha	2	1999.00	1	1999.00
Sasha	3	3999.00	2	7998.00

3

Объедини  
м по  
user\_name  
(group by)

```
1 • use shop;
2
3 • SELECT `order` .user_name, sum(price * `count`) as total_price FROM `order`
4
5     inner join order_products on order_products.order_id = `order` . id
6
7     inner join products on products.id = order_products.product_id
8     group by `order` . user_name;
9
10
```

100% 69:5

Result Grid Filter Rows: Search Export:

user_name	total_price
Petya	9996.00
Sasha	13995.00
Vasya	11996.00

use shop;  
**SELECT `order` .user\_name, sum(price \* `count`) as total\_price FROM `order`  
inner join order\_products on order\_products.order\_id = `order` . id  
inner join products on products.id = order\_products.product\_id  
group by `order` . user\_name;**

4

Объедини  
м с  
помощью  
group by по  
user name,  
min/max  
price,  
количество  
товаров в  
корзине,  
Итого  
сумма в  
корзине

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a text editor window containing the following SQL code:

```

1 • use shop;
2 • SELECT `order` .user_name, max(price), min(price), sum(`count`), sum(price * `count`) as total_price FROM `order`
3   inner join order_products on order_products.order_id = `order` . id
4   inner join products on products.id = order_products.product_id
5   group by `order` . user_name;
6
7
8
9
10

```

The code uses three tables: `order`, `order\_products`, and `products`. It selects the user name, maximum price, minimum price, sum of counts, and the total price (sum of price times count) for each user. It joins the `order` table with the `order\_products` table on the `order\_id` column, and the `order\_products` table with the `products` table on the `product\_id` column. The results are grouped by the user name.

Below the code is a result grid showing the output:

user_name	max(price)	min(price)	sum(`count`)	total_price
Petya	3999.00	1999.00	4	9996.00
Sasha	3999.00	1999.00	5	13995.00
Vasya	3999.00	1999.00	4	11996.00

**use shop;**  
**SELECT `order` .user\_name, max(price), min(price), sum(`count`), sum(price \* `count`) as total\_price FROM `order`**  
**inner join order\_products on order\_products.order\_id = `order` . id**  
**inner join products on products.id = order\_products.product\_id**  
**group by `order` . user\_name;**

19

Ограничения для агрегированных данных HAVING.  
ИНДЕКСЫ для оптимизации работы DataBase

№ Действие

Команда на языке SQL и скрин результата

1

Ограничен  
ия для  
агрегирова  
нных  
данных  
HAVING.

```
1 • use shop;
2 • SELECT `order` .user_name, max(price), min(price), sum(`count`), sum(price * `count`) as total_price FROM `order`
3     inner join order_products on order_products.order_id = `order` . id
4
5     inner join products on products.id = order_products.product_id
6     group by `order` . user_name
7
8     having total_price >= 10000;
```

100%

5:9

Result Grid

Filter Rows:

Search

Export:

user_name	max(price)	min(price)	sum(`count`)	total_price
Sasha	3999.00	1999.00	5	13995.00
Vasya	3999.00	1999.00	4	11996.00

use shop;

```
SELECT `order` .user_name, max(price), min(price), sum(`count`), sum(price * `count`) as total_price FROM `order`
inner join order_products on order_products.order_id = `order` . id
inner join products on products.id = order_products.product_id
group by `order` . user_name
having total_price >= 10000;
```

2

Индекс и производительность

Для ускорения производительности можно использовать индексацию.

3

Предположим база увеличилась

Для обработки такого запроса ушло 0,016sec

The screenshot shows the MySQL Workbench interface with the following details:

- Tab Bar:** Урок 18. Group By, product, product - Table
- Query Editor:** Contains two queries:
  1. SELECT count(\*) from shop.product ;
  2. SELECT \* FROM shop.product where price = 5042;
- Result Grid:** Displays the results of the second query:

	id	brand_id	product_type_id	category_id	price
▶	1050	2	1	2	5042.00
	3051	2	1	2	5042.00
*	NULL	NULL	NULL	NULL	NULL

4

Добавим  
индекс в  
таблицу  
Products-  
Alterable-  
Indexes

Table Name: product Schema: shop  
Collation: utf8 - utf8\_bin Engine: InnoDB  
Comments:

Index Name	Type
PRIMARY	PRIMARY
fk_brand_product_idx	INDEX
fk_category_product_idx	INDEX
fk_product_type_product_idx	INDEX
price_index	INDEX

Column	#	Order	Length
id		ASC	
brand_id		ASC	
product_type_id		ASC	
category_id		ASC	
price	1	ASC	

Storage Type: Key Block Size: 0 Parser: Index Comment:

Columns Indexes Foreign Keys Triggers Partitioning Options

Apply Revert Cont

Теперь для обработки запроса из пункта 3 уйдет 0,000 сек

20

## Транзакции, на примере банковского перевода от одного Usera другому

№ Действие

## Команда на языке SQL и скрин результата

1

Создаем  
Таблицу  
USER\_BAN  
K\_ACCOUNT

The screenshot shows the MySQL Workbench interface for creating a table. The table name is 'user\_bank\_account' and it is being created in the 'shop' schema.

**Table Structure:**

Column	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	G	Default / Expression
id	INT	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	
money	DECIMAL(10,2)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					
user_name	VARCHAR(45)	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>					

**Column Details for 'user\_name':**

Column Name:	user_name	Datatype:	VARCHAR(45)
Collation:	Table Default	Algorithm:	Default
Comments:			
1	CREATE TABLE `shop`.`user_bank_account` (		
2	`id` INT NOT NULL,		
3	`money` DECIMAL(10,2) NOT NULL,		
4	`user_name` VARCHAR(45) NOT NULL,		
5	PRIMARY KEY (`id`));		
6			

**Buttons at the bottom:**

- Columns
- Indexes
- Foreign Keys
- Triggers
- Partitioning
- Options
- Apply
- Revert

2

Добавим  
значения в  
таблицу  
USER\_BAN  
K\_ACCOUNT

id	money	user_name
1	100.00	Dima
2	200.00	Eygen
NULL	NULL	NULL

```
insert into `shop`.`user_bank_account` (id, money, user_name) values ('1',  
'100', 'Dima');  
insert into `shop`.`user_bank_account` (id, money, user_name) values ('2',  
'200', 'Eygen');  
  
select * from `shop`.`user_bank_account`;
```

3

Сделаем транзакци ю перевода 200 рублей от Димы Евгению.

```
use shop;  
start transaction;  
update user_bank_account set money = money - 200 where id = 1;  
update user_bank_account set money = money + 200 where id = 2;  
COMMIT;  
  
select * from `shop`.`user_bank_account`;
```

The screenshot shows a MySQL Workbench interface. At the top, there's a toolbar with various icons. Below it is a text editor window displaying a SQL script. The script starts with 'use shop;', followed by a transaction block: 'start transaction;', 'update user\_bank\_account set money = money - 200 where id = 1;', 'update user\_bank\_account set money = money + 200 where id = 2;', and 'COMMIT;'. After the transaction block, there is a 'select \* from `shop`.`user\_bank\_account`;' statement. The text editor has line numbers on the left and a status bar at the bottom indicating '100%' and '1:7'. Below the text editor is a 'Result Grid' tab which is currently selected. It displays a table with three columns: 'id', 'money', and 'user\_name'. The data shows two rows: one for user\_id 1 with money -100.00 and user\_name Dima, and one for user\_id 2 with money 400.00 and user\_name Egen. There are also three empty rows below them labeled 'NULL'.

id	money	user_name
1	-100.00	Dima
2	400.00	Eygen
NULL	NULL	NULL
NULL	NULL	NULL
NULL	NULL	NULL

# ACID

## СВОЙСТВА ТРАНЗАКЦИЙ

1. **Atomicity** — Атомарность
2. **Consistency** — Согласованность
3. **Isolation** — Изолированность
4. **Durability** — Долговечность

Запись этих двух операций в совокупности будет гарантировать нам одновременное свершение транзакции, либо не выполнится ни одна в случае проблемы.