

Реляционные базы данных: **SQL. Часть 1.**



Роман
Гордиенко



Роман Гордиенко

Backend Developer, Factory5



[Роман Гордиенко](#)

План занятия

1. [Данные](#)
2. [Простые запросы](#)
3. [Работа с числами](#)
4. [Работа со строками](#)
5. [Работа с датами и временем](#)
6. [BETWEEN](#)
7. [Итоги](#)
8. [Домашнее задание](#)



Данные

Данные

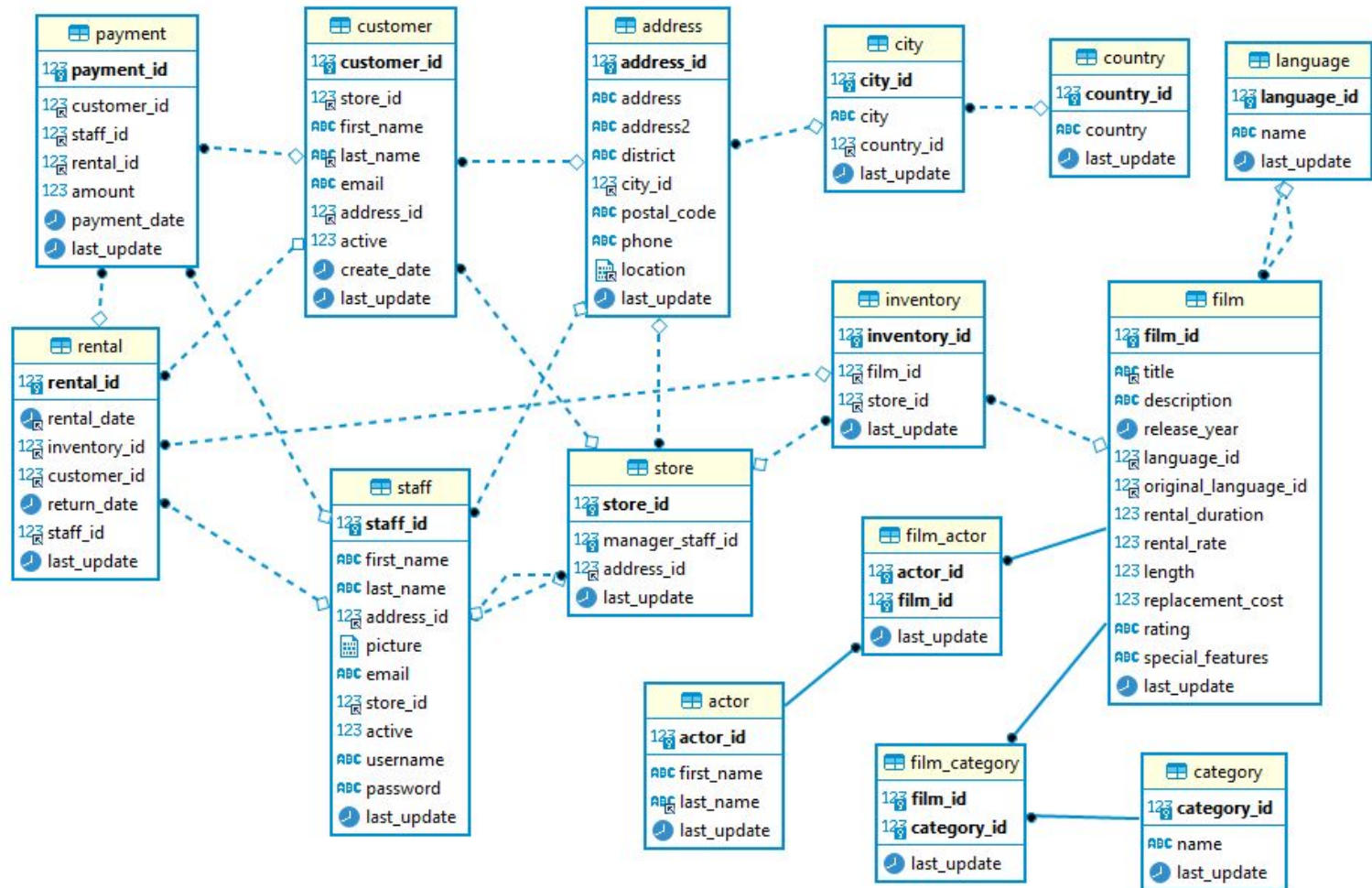
На прошлом занятии вы должны были установить подключение к базе данных MySQL и начать работать с данными.

Напомним [ссылку](#) на дамп файл с учебной базой данных.

Данный dataset состоит из:

- 16 таблиц,
- 7 представлений,
- 6 хранимых процедур.

Данные. ER-диаграмма





Простые запросы

Простые запросы. SELECT и FROM

Для того чтобы получить данные, в запросе нужно указать:

- из какой таблицы хотим получить данные – предложение **FROM**;
- какие данные хотим вывести в результат – предложение **SELECT**.

```
SELECT * FROM customer;
```

Нужно вывести всех пользователей из таблицы customer:

customer_id	store_id	first_name	last_name	email	address_id	active	create_date	last_update
1	1	MARY	SMITH	MARY.SMITH@sakilacustomer.org	5	1	2006-02-14 22:04:36	2006-02-15 04:57:20
2	1	PATRICIA	JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	6	1	2006-02-14 22:04:36	2006-02-15 04:57:20
3	1	LINDA	WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	7	1	2006-02-14 22:04:36	2006-02-15 04:57:20
4	2	BARBARA	JONES	BARBARA.JONES@sakilacustomer.org	8	1	2006-02-14 22:04:36	2006-02-15 04:57:20
5	1	ELIZABETH	BROWN	ELIZABETH.BROWN@sakilacustomer.org	9	1	2006-02-14 22:04:36	2006-02-15 04:57:20

Простые запросы. SELECT и FROM

Оператор * означает, что хотим вывести в результат все столбцы из таблицы customer, если нужно вывести определенные столбцы, то их нужно перечислить:

```
SELECT customer_id, last_name, first_name FROM customer;
```

customer_id	last_name	first_name
1	SMITH	MARY
2	JOHNSON	PATRICIA
3	WILLIAMS	LINDA
4	JONES	BARBARA
5	BROWN	ELIZABETH
6	DAVIS	JENNIFER
7	MILLER	MARIA

Простые запросы. ALIAS

Алиасы (псевдонимы) нужны для того, чтобы задавать временные названия для столбцов и таблиц. Алиасы для столбцов позволяют дать понятные названия для вычисляемых значений, а краткие алиасы для таблиц позволяют упростить написание запросов.

Для задания алиасов используется оператор **AS**, но его можно опускать.

Чтобы не писать полные названия таблиц, зададим в запросе на следующем слайде краткие алиасы и для вычисляемого столбца также зададим временное имя с обозначением результата.

Простые запросы. ALIAS

```
SELECT f.title, c.name, f.rental_rate/f.rental_duration AS cost_per_day
FROM film f
JOIN film_category fc ON fc.film_id = f.film_id
JOIN category c ON c.category_id = fc.category_id;
```

title	name	cost_per_day
AMADEUS HOLY	Action	0.165000
AMERICAN CIRCUS	Action	1.663333
ANTITRUST TOMATOES	Action	0.598000
ARK RIDGEMONT	Action	0.165000
BAREFOOT MANCHURIAN	Action	0.498333
BERETS AGENT	Action	0.598000
BRIDE INTRIGUE	Action	0.141429
BULL SHAWSHANK	Action	0.165000
CADDYSHACK JEDI	Action	0.330000
CAMPUS REMEMBER	Action	0.598000

Простые запросы. ORDER BY

В независимости от того, в каком порядке данные хранятся в базе данных, SQL возвращает результат в непредсказуемом порядке. Чтобы явно задать порядок сортировки, используется оператор **ORDER BY**. Для того чтобы задать направление сортировки, нужно указывать **ASC** – от меньшего к большему (по умолчанию) или **DESC** – от большего к меньшему.

Давайте возьмем запрос с получением стоимости аренды фильма за день и отсортируем по стоимости аренды за день от большего к меньшему, а потом по названию фильма.

Простые запросы. ORDER BY

```
SELECT title, rental_rate/rental_duration AS cost_per_day
FROM film
ORDER BY cost_per_day DESC, title;
```

title	cost_per_day
ACE GOLDFINGER	1.663333
AMERICAN CIRCUS	1.663333
AUTUMN CROW	1.663333
BACKLASH UNDEFEATED	1.663333
BEAST HUNCHBACK	1.663333
BEHAVIOR RUNAWAY	1.663333
BILKO ANONYMOUS	1.663333
CARIBBEAN LIBERTY	1.663333
CASPER DRAGONFLY	1.663333

Простые запросы. LIMIT и OFFSET

Если нужно получить первые N записей из результата, используется оператор **LIMIT**.

Если нужно исключить из результата первые N записей, используется оператор **OFFSET**.

Возьмем предыдущий запрос и получим первые 10 записей начиная с 58:

Простые запросы. LIMIT и OFFSET

```
SELECT title, rental_rate/rental_duration AS cost_per_day
FROM film
ORDER BY cost_per_day DESC, title
LIMIT 10
OFFSET 57;
```

title	cost_per_day
TYCOON GATHERING	1.663333
VELVET TERMINATOR	1.663333
VIRTUAL SPOILERS	1.663333
WIFE TURN	1.663333
ZORRO ARK	1.663333
ALI FOREVER	1.247500
AMELIE HELLFIGHTERS	1.247500
BIRCH ANTITRUST	1.247500
BRANNIGAN SUNRISE	1.247500
BRIGHT ENCOUNTERS	1.247500

Простые запросы. DISTINCT

Для получения уникальных значений в результате, используется оператор **DISTINCT**.

К примеру, нужно получить уникальный список имен пользователей:

```
SELECT DISTINCT first_name  
FROM customer;
```

first_name
MARY
PATRICIA
LINDA
BARBARA
ELIZABETH
JENNIFER
MARIA
SUSAN

Простые запросы. DISTINCT

Если нужно получить уникальные значения по нескольким столбцам, то данные столбцы перечисляются после оператора **DISTINCT**:

```
SELECT DISTINCT last_name, first_name
FROM customer;
```

last_name	first_name
SMITH	MARY
JOHNSON	PATRICIA
WILLIAMS	LINDA
JONES	BARBARA
BROWN	ELIZABETH
DAVIS	JENNIFER
MILLER	MARIA
WILSON	SUSAN
MOORE	MARGARET
TAYLOR	DOROTHY

Простые запросы. WHERE

Как правило, при получении данных нужно указать условия, по которым нужно их отфильтровать, для этого используется оператор **WHERE**. Если условий нужно использовать несколько, то используются логические операторы **AND** и **OR**.

Для отрицания в условии используется оператор **NOT**.

Выведем в результат платежи:

- более 7 у.е. и которые совершил сотрудник с идентификатором равным 2,
- менее 5 у.е. и которые совершил сотрудник с идентификатором равным 1.

Простые запросы. WHERE

Обратите внимание, что оператор **AND** имеет приоритет перед **OR**

```
SELECT *  
FROM payment  
WHERE amount > 7 AND staff_id = 2 OR amount < 5 AND staff_id = 1;
```

payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
1	1	1	76	2.99	2005-05-25 11:30:37	2006-02-15 22:12:30
2	1	1	573	0.99	2005-05-28 10:35:23	2006-02-15 22:12:30
5	1	2	1476	9.99	2005-06-15 21:08:46	2006-02-15 22:12:30
6	1	1	1725	4.99	2005-06-16 15:18:57	2006-02-15 22:12:30
7	1	1	2308	4.99	2005-06-18 08:41:48	2006-02-15 22:12:30

Простые запросы. CAST

При работе с разными типами данных часто нужно преобразовывать один тип данных к другому, для этого используется оператор **CAST** со следующим синтаксисом:

```
CAST(value AS type)
```

В таблице payment столбец payment_date имеет тип данных datetime, то есть дата и время, а нужно работать только с датой, для этого преобразуем datetime к date:

```
SELECT payment_id, CAST(payment_date AS DATE)
FROM payment;
```

payment_id	payment_date
1	2005-05-25 11:30:37
2	2005-05-28 10:35:23
3	2005-06-15 00:54:12

payment_id	CAST(payment_date AS DATE)
1	2005-05-25
2	2005-05-28
3	2005-06-15

Таблица приведения типов, на примере MSSQL

	To	binary	varbinary	char	nchar	nvarchar	datetime	smalldatetime	date	time	datetimeoffset	datetime2	decimal	numeric	float	real	bigint	int(INT4)	smallint(INT2)	tinyint(INT1)	money	smallmoney	bit	timestamp	uniqueidentifier	image	ntext	text	sql_variant	xml	CLR UDT	hierarchyid
From																																
binary																																
varbinary																																
char																																
varchar																																
nchar																																
nvarchar																																
datetime																																
smalldatetime																																
date																																
time																																
datetimeoffset																																
datetime2																																
decimal																																
numeric																																
float																																
real																																
bigint																																
int(INT4)																																
smallint(INT2)																																
tinyint(INT1)																																
money																																
smallmoney																																
bit																																
timestamp																																
uniqueidentifier																																
image																																
ntext																																
text																																
sql_variant																																
xml																																
CLR UDT																																
hierarchyid																																

■ Explicit conversion

● Implicit conversion

✗ Conversion not allowed

◆ Requires explicit CAST to prevent the loss of precision or scale that might occur in an implicit conversion.

○ Implicit conversions between xml data types are supported only if the source or target is untyped xml. Otherwise, the conversion must be explicit.



Работа с числами

Округление

Для округления в MySQL используются следующие функции:

- **ROUND** – округляет число до заданного числа десятичных знаков,
- **TRUNCATE** – усекает число до указанного числа десятичных знаков,
- **FLOOR** – возвращает наибольшее целочисленное значение, которое меньше или равно числу,
- **CEIL** – возвращает наименьшее целочисленное значение, которое больше или равно числу,
- **ABS** – возвращает абсолютное (положительное) значение числа.

Округление

Округлим значения используя разные функции:

```
SELECT ROUND(100.576); -- 101
SELECT ROUND(100.576, 2); -- 100.58
SELECT TRUNCATE(100.576, 2); -- 100.57
SELECT FLOOR(100.576); -- 100
SELECT CEIL(100.576); -- 101
SELECT ABS(-100.576); -- 100.576
```

Получим «красивый» результат стоимости аренды за день:

```
SELECT title, ROUND(rental_rate/rental_duration, 2) AS cost_per_day
FROM film
ORDER BY cost_per_day DESC, title;
```

title	cost_per_day
ACE GOLDFINGER	1.66
AMERICAN CIRCUS	1.66
AUTUMN CROW	1.66
BACKLASH UNDEFEATED	1.66
BEAST HUNCHBACK	1.66

Арифметические операторы

SQL поддерживает все основные арифметические операторы:

- $+$ $-$ $*$ $/$ – стандартные операторы,
- **POWER** – возведение в степень,
- **SQRT** – возвращает квадратный корень числа,
- **COS, SIN, TAN, COT, etc** – геометрические операторы,
- **DIV** – целочисленное деление,
- **%** – остаток от деления,
- **GREATEST/LEAST** – возвращает наибольшее/наименьшее значение из списка,
- **RAND** – возвращает случайное число в диапазоне от 0 (включительно) до 1 (исключительно).

Арифметические операторы

Посмотрим на работу некоторых функций:

```
SELECT POWER(2, 3); -- 8
SELECT SQRT(64); -- 8
SELECT 64 DIV 6; -- 10
SELECT 64%6; -- 4
SELECT GREATEST(17, 5, 18, 21, 16); -- 21
SELECT LEAST(17, 5, 18, 21, 16); -- 5
SELECT RAND(); -- 0.005757967015502944
```

Арифметические операторы

Посмотрим на работу некоторых функций на данных:

```
SELECT rental_rate, rental_duration,  
       rental_rate + rental_duration a,  
       rental_rate - rental_duration b,  
       rental_rate * rental_duration c,  
       rental_rate / rental_duration d,  
       rental_rate % rental_duration e,  
       rental_rate DIV rental_duration f,  
       POWER(rental_rate, rental_duration) g,  
       COS(rental_rate) h, SIN(rental_duration) j  
FROM film;
```

rental_rate	rental_duration	a	b	c	d	e	f	g	h	j	
0.99	6	6.99	-5.01	5.94	0.165000	0.99	0	0.941480149401	0.5486898605815875	-0.27941549819892586	
4.99	3	7.99	1.99	14.97	1.663333	1.99	1	124.25149900000001	0.27405891954542744	0.1411200080598672	
2.99	7	9.99	-4.01	20.93	0.427143	2.99	0	2136.477474431122	-0.988531820827396	0.6569865987187891	
2.99	5	7.99	-2.01	14.95	0.598000	2.99	0	238.97691014990008	-0.988531820827396	-0.9589242746631385	
2.99	6	8.99	-3.01	17.94	0.498333	2.99	0	714.5409613482013	-0.988531820827396	-0.27941549819892586	
2.99	3	5.99	-0.01	8.97	0.996667	2.99	0	26.730899000000004	-0.988531820827396	0.1411200080598672	
4.99	6	10.99	-1.01	29.94	0.831667	4.99	0	15438.435003747005	0.27405891954542744	-0.27941549819892586	



Работа со строками

Работа со строками

Разберем основные функции для работы с подстроками и строками:

- **CONCAT, CONCAT_WS** – соединяет строки в одну, **_WS** – по сепаратору,
- **LENGTH** – возвращает длину строки в байтах,
- **CHAR_LENGTH** – возвращает длину строки в символах,
- **POSITION** – возвращает позицию первого вхождения подстроки в строку,
- **SUBSTR** – извлекает подстроку из строки,

Работа со строками

- **LEFT / RIGHT** – извлекает ряд символов из строки начиная слева / справа,
- **LOWER / UPPER** – преобразует строку в нижний / верхний регистр,
- **INSERT** – вставляет подстроку в строку в указанной позиции и для определенного количества символов,
- **TRIM** – удаляет начальные и конечные пробелы из строки,
- **REPLACE** – заменяет все вхождения подстроки в строке на новую подстроку,
- **SUBSTRING_INDEX** – возвращает подстроку строки до того, как появится указанное число разделителей.

Работа со строками

Давайте разберем, как эти функции работают на практике:

```
SELECT CONCAT(last_name, ' ', first_name, ' ', email) FROM customer;  
SELECT CONCAT_WS(' ', last_name, first_name, email) FROM customer;
```

```
CONCAT_WS(' ', last_name, first_name, email)  
-----+  
SMITH MARY MARY.SMITH@sakilacustomer.org  
JOHNSON PATRICIA PATRICIA.JOHNSON@sakilacustomer.org  
WILLIAMS LINDA LINDA.WILLIAMS@sakilacustomer.org
```

```
SELECT    LENGTH(last_name), CHAR_LENGTH(last_name),  
          LENGTH('Привет'), CHAR_LENGTH('Привет')  
FROM customer;
```

```
LENGTH(last_name)|CHAR_LENGTH(last_name)|LENGTH('Привет')|CHAR_LENGTH('Привет')|  
-----+-----+-----+-----+  
5|5|12|6|  
4|4|12|6|  
5|5|12|6|
```

Работа со строками

```
SELECT  POSITION('D' IN last_name), SUBSTR(last_name, 2, 3),
        LEFT(last_name, 3), RIGHT(last_name, 3)
FROM customer;
```

POSITION('D' IN last_name)	SUBSTR(last_name, 2, 3)	LEFT(last_name, 3)	RIGHT(last_name, 3)
0	BNE	ABN	NEY
2	DAM	ADA	DAM
2	DAM	ADA	AMS
7	LEX	ALE	DER

```
SELECT  LOWER(last_name), INSERT(last_name, 'MAX', 1, 5),
        REPLACE(last_name, 'A', 'X')
FROM customer;
```


LOWER(last_name)	INSERT(last_name, 'MAX', 1, 5)	REPLACE(last_name, 'A', 'X')
abney	ABNEY	XBNEY
adam	ADAM	XDXM
adams	ADAMS	XDXMS
alexander	ALEXANDER	XLEXXNDER

Работа со строками

Выражение **LIKE** возвращает **true**, если строка соответствует заданному шаблону. Выражение **NOT LIKE** возвращает **false**, когда **LIKE** возвращает **true** и наоборот. Если шаблон не содержит знаков процента и подчеркиваний, тогда шаблон представляет в точности строку и **LIKE** работает как оператор сравнения. Подчеркивание (**_**) в шаблоне подменяет (вместо него подходит) любой символ. Знак процента (**%**) подменяет любую (в том числе и пустую) последовательность символов.

```
SELECT CONCAT(last_name, ' ', first_name)
FROM customer
WHERE first_name LIKE '%jam%';
```

```
concat(last_name, ' ', first_name)|
-----+
RICE JAMIE
GANNON JAMES
VARNEY BENJAMIN
WAUGH JAMIE
```



Работа с датами и временем

Работа с датами и временем

Разберем основные функции для работы с датами и временем:

- **NOW / CURDATE** – возвращает текущие дату и время / дату,
- **DATE_ADD** – добавляет интервал времени/даты к дате, а затем возвращает дату, работает как с датой, так и со временем,
- **DATE_SUB** – вычитает интервал времени/даты из даты, а затем возвращает дату, работает как с датой, так и со временем,
- **YEAR / MONTH / DAY** – возвращает год / месяц / день месяца для заданной даты,
- **EXTRACT** – извлекает часть из заданной даты,

Работа с датами и временем

- **DATEDIFF** – возвращает количество дней между двумя значениями даты,
- **QUARTER** – возвращает квартал года для заданного значения даты,
- **DATE_FORMAT** – форматирует указанную дату,
- **TIME_FORMAT** – форматирует время по заданному формату,
- **DATE** – извлекает дату из выражения datetime.

Работа с датами и временем

Время практики!

```
SELECT DATE_ADD(NOW(), INTERVAL 3 DAY);
```

now()	DATE_ADD(now(), INTERVAL 3 day)
2021-08-30 13:48:10	2021-09-02 13:48:10

```
SELECT DATE_SUB(CURDATE(), INTERVAL 3 DAY);
```

CURDATE()	DATE_SUB(CURDATE(), INTERVAL 3 day)
2021-08-30	2021-08-27

```
SELECT YEAR(NOW()), MONTH(NOW()), WEEK(NOW()), DAY(NOW());
```

YEAR(NOW())	MONTH(NOW())	WEEK(NOW())	DAY(NOW())
2021	8	35	30

Работа с датами и временем

```
SELECT EXTRACT(HOUR FROM NOW()), EXTRACT(DAY_MINUTE FROM NOW()),  
       EXTRACT(DAY FROM NOW());
```

```
EXTRACT(HOUR FROM NOW())|EXTRACT(DAY_MINUTE FROM NOW())|EXTRACT(DAY FROM NOW())|  
-----+-----+-----+  
13|1352|30|
```

```
SELECT DATEDIFF(return_date, rental_date), QUARTER(return_date) FROM rental;
```

```
DATEDIFF(return_date, rental_date)|QUARTER(return_date)|  
-----+-----+  
2|2|  
4|2|  
8|2|  
10|2|
```

```
SELECT DATE_FORMAT(payment_date, '%D - %A - %Y'),  
       TIME_FORMAT(TIME(payment_date), '%R') FROM payment;
```

```
DATE_FORMAT(payment_date, '%D - %a - %Y')|TIME_FORMAT(TIME(payment_date), '%r')|  
-----+-----+  
25th - Wed - 2005|11:30:37 AM|  
28th - Sat - 2005|10:35:23 AM|  
15th - Wed - 2005|12:54:12 AM|
```

Работа с датами и временем

При работе с датой и временем нужно помнить: если из **DATE** сделать **DATETIME**, то это будет дата с нулевым временем.

К примеру, есть дата '2020-01-01', если с ней работать, как с **DATETIME**, то **SQL** будет эту дату воспринимать, как '2020-01-01 00:00:00'.

Соответственно, если значения в БД хранятся в **DATETIME**, но в запросе нужно работать именно с датами, то нужно явно приводить значения к нужному типу данных. Причем это касается не только дат, но и остальных типов данных.



BETWEEN

BETWEEN

Для того чтобы найти значения в заданном диапазоне, используется оператор **BETWEEN**. Данный оператор можно использовать с числами, строками и датами. Крайние значения включаются в результат.

К примеру, нужно найти все платежи, стоимость которых между 5 и 7 включительно:

```
SELECT * FROM payment WHERE amount BETWEEN 5 AND 7;
```

payment_id	customer_id	staff_id	rental_id	amount	payment_date	last_update
3	1	1	1185	5.99	2005-06-15 00:54:12	2006-02-15 22:12:30
10	1	2	4526	5.99	2005-07-08 03:17:05	2006-02-15 22:12:30
11	1	1	4611	5.99	2005-07-08 07:33:56	2006-02-15 22:12:30



Итоги

Итоги

В данной лекции мы:

- Научились писать простые запросы;
- Разобрали функции для работы с числовыми типами данных;
- Разобрали функции для работы со строковыми типами данных;
- Разобрали функции для работы с датой и временем.





Домашнее задание

Домашнее задание

Давайте посмотрим ваше [домашнее задание](#).

- Вопросы по домашней работе задавайте **в чате** мессенджера Slack.
- Задачи можно сдавать **по частям**.
- Зачёт по домашней работе проставляется после того, как **приняты все задачи**.

**Задавайте вопросы и
пишите ОТЗЫВ о лекции!**

Роман Гордиенко

 [Роман Гордиенко](#)