# **Соединение таблиц**

## **Неявное соединение таблиц**

Нередко возникает необходимость в одном запросе получить данные сразу из нескольких таблиц. Для сведения данных из разных таблиц мы можем использовать разные способы. В данной статье рассмотрим не самый распространенный, однако довольно простой способ, который представляет неявное соединение таблиц.

Допустим, у нас есть следующие таблицы, которые связаны между собой связями:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | CREATE TABLE Products  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  ProductName VARCHAR(30) NOT NULL,  Manufacturer VARCHAR(20) NOT NULL,  ProductCount INT DEFAULT 0,  Price DECIMAL NOT NULL  );  CREATE TABLE Customers  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  FirstName VARCHAR(30) NOT NULL  );  CREATE TABLE Orders  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  ProductId INT NOT NULL,  CustomerId INT NOT NULL,  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price DECIMAL NOT NULL,  FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE,  FOREIGN KEY (CustomerId) REFERENCES Customers(Id) ON DELETE CASCADE  ); |

Здесь таблицы Products и Customers связаны с таблицей Orders связью один ко многим. Таблица Orders в виде внешних ключей ProductId и CustomerId содержит ссылки на столбцы Id из соответственно таблиц Products и Customers. Также она хранит количество купленного товара (ProductCount) и и по какой цене он был куплен (Price). И кроме того, таблицы также хранит в виде столбца CreatedAt дату покупки.

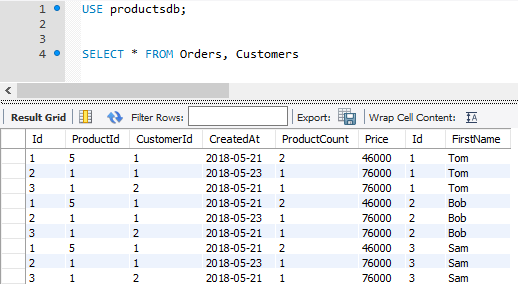
Пусть эти таблицы будут содержать следующие данные:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | INSERT INTO Products (ProductName, Manufacturer, ProductCount, Price)  VALUES ('iPhone X', 'Apple', 2, 76000),  ('iPhone 8', 'Apple', 2, 51000),  ('iPhone 7', 'Apple', 5, 42000),  ('Galaxy S9', 'Samsung', 2, 56000),  ('Galaxy S8', 'Samsung', 1, 46000),  ('Honor 10', 'Huawei', 2, 26000),  ('Nokia 8', 'HMD Global', 6, 38000);    INSERT INTO Customers(FirstName) VALUES ('Tom'), ('Bob'),('Sam');    INSERT INTO Orders (ProductId, CustomerId, CreatedAt, ProductCount, Price)  VALUES  (  (SELECT Id FROM Products WHERE ProductName='Galaxy S8'),  (SELECT Id FROM Customers WHERE FirstName='Tom'),  '2018-05-21',  2,  (SELECT Price FROM Products WHERE ProductName='Galaxy S8')  ),  (  (SELECT Id FROM Products WHERE ProductName='iPhone X'),  (SELECT Id FROM Customers WHERE FirstName='Tom'),  '2018-05-23',  1,  (SELECT Price FROM Products WHERE ProductName='iPhone X')  ),  (  (SELECT Id FROM Products WHERE ProductName='iPhone X'),  (SELECT Id FROM Customers WHERE FirstName='Bob'),  '2018-05-21',  1,  (SELECT Price FROM Products WHERE ProductName='iPhone X')  ); |

Теперь соединим две таблицы Orders и Customers:

|  |  |
| --- | --- |
| 1 | SELECT \* FROM Orders, Customers; |

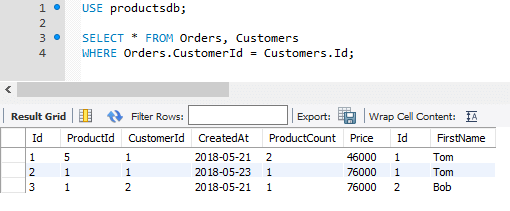
При такой выборке каждая строка из таблицы Orders будет соединяться с каждой строкой из таблицы Customers. То есть, получится перекрестное соединение. Например, в Orders три строки, а в Customers то же три строки, значит мы получим 3 \* 3 = 9 строк:



Но вряд ли это тот результат, который хотелось бы видеть. Тем более каждый заказ из Orders связан с конкретным покупателем из Customers, а не со всеми возможными покупателями.

Чтобы решить задачу, необходимо использовать выражение **WHERE** и фильтровать строки при условии, что поле CustomerId из Orders соответствует полю Id из Customers:

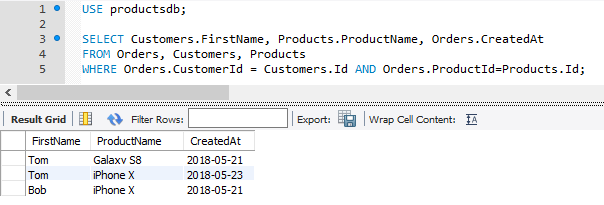
|  |  |
| --- | --- |
| 1  2 | SELECT \* FROM Orders, Customers  WHERE Orders.CustomerId = Customers.Id; |



Теперь объединим данные по трем таблицам Orders, Customers и Proucts. То есть получим все заказы и добавим информацию по клиенту и связанному товару:

|  |  |
| --- | --- |
| 1  2  3 | SELECT Customers.FirstName, Products.ProductName, Orders.CreatedAt  FROM Orders, Customers, Products  WHERE Orders.CustomerId = Customers.Id AND Orders.ProductId=Products.Id; |

Так как здесь нужно соединить три таблицы, то применяются как минимум два условия. Ключевой таблицей остается Orders, из которой извлекаются все заказы, а затем к ней подсоединяется данные по клиенту по условию Orders.CustomerId = Customers.Id и данные по товару по условию Orders.ProductId=Products.Id



В данном случае названия таблиц сильно увеличивают код, но мы его можем сократить за счет использования псевдонимов таблиц:

|  |  |
| --- | --- |
| 1  2  3 | SELECT C.FirstName, P.ProductName, O.CreatedAt  FROM Orders AS O, Customers AS C, Products AS P  WHERE O.CustomerId = C.Id AND O.ProductId=P.Id; |

Если необходимо при использовании псевдонима выбрать все столбцы из определенной таблицы, то можно использовать звездочку:

|  |  |
| --- | --- |
| 1  2  3 | SELECT C.FirstName, P.ProductName, O.\*  FROM Orders AS O, Customers AS C, Products AS P  WHERE O.CustomerId = C.Id AND O.ProductId=P.Id; |

## **Inner Join**

В прошлой теме было рассмотрено неявное соединение таблиц с помощью простой выборки путем сведения данных. Но, как правило, более распространенный подход соединения данных из разных таблиц представляет применение оператора **JOIN**. Общий формальный синтаксис применения оператора INNER JOIN:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT столбцы  FROM таблица1  [INNER] JOIN таблица2  ON условие1  [[INNER] JOIN таблица3  ON условие2] |

После оператора **JOIN** идет название второй таблицы, из которой надо добавить данные в выборку. Перед JOIN может использоваться необязательное ключевое слово **INNER**. Его наличие или отсутствие ни на что не влияет. Затем после ключевого слова **ON** указывается условие соединения. Это условие устанавливает, как две таблицы будут сравниваться. В большинстве случаев для соединения применяется первичный ключ главной таблицы и внешний ключ зависимой таблицы.

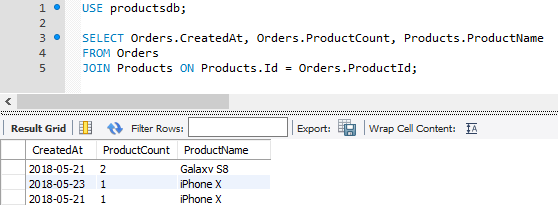
Возьмем таблицы с данными из прошлой темы:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24 | CREATE TABLE Products  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  ProductName VARCHAR(30) NOT NULL,  Manufacturer VARCHAR(20) NOT NULL,  ProductCount INT DEFAULT 0,  Price DECIMAL NOT NULL  );  CREATE TABLE Customers  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  FirstName VARCHAR(30) NOT NULL  );  CREATE TABLE Orders  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  ProductId INT NOT NULL,  CustomerId INT NOT NULL,  CreatedAt DATE NOT NULL,  ProductCount INT DEFAULT 1,  Price DECIMAL NOT NULL,  FOREIGN KEY (ProductId) REFERENCES Products(Id) ON DELETE CASCADE,  FOREIGN KEY (CustomerId) REFERENCES Customers(Id) ON DELETE CASCADE  ); |

Используя JOIN, выберем все заказы и добавим к ним информацию о товарах:

|  |  |
| --- | --- |
| 1  2  3 | SELECT Orders.CreatedAt, Orders.ProductCount, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId; |

Поскольку таблицы могут содержать столбцы с одинаковыми названиями, то при указании столбцов для выборки указывается их полное имя вместе с именем таблицы, например, "Orders.ProductCount".

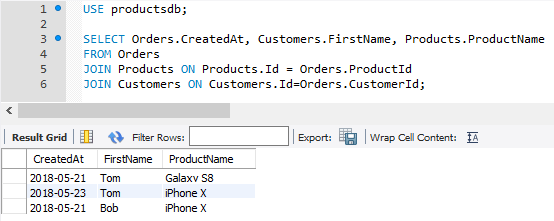


Используя псевдонимы для таблиц, можно сократить код:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT O.CreatedAt, O.ProductCount, P.ProductName  FROM Orders AS O  JOIN Products AS P  ON P.Id = O.ProductId; |

Также можно присоединять данные сразу из нескольких таблиц. Например, добавим к заказу информацию о покупателе из таблицы Customers:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId  JOIN Customers ON Customers.Id=Orders.CustomerId; |



Благодаря соединению таблиц мы можем использовать их столбцы для фильтрации выборки или ее сортировки:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId  JOIN Customers ON Customers.Id=Orders.CustomerId  WHERE Products.Price > 45000  ORDER BY Customers.FirstName; |

Условия после ключевого слова ON могут быть более сложными по составу:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT Orders.CreatedAt, Customers.FirstName, Products.ProductName  FROM Orders  JOIN Products ON Products.Id = Orders.ProductId AND Products.Manufacturer='Apple'  JOIN Customers ON Customers.Id=Orders.CustomerId  ORDER BY Customers.FirstName; |

В данном случае выбираем все заказы на товары, производителем которых является Apple.

При использовании оператора JOIN следует учитывать, что процесс соединения таблиц может быть ресурсоемким, поэтому следует соединять только те таблицы, данные из которых действительно необходимы. Чем больше таблиц соединяется, тем больше снижается производительность.

## **Left / Right Join**

В предыдущей теме рассматривля Inner Join или внутреннее соединение таблиц. Но также в MySQL мы можем использовать и так называемое внешнее соединение или Outer Join. В отличие от Inner Join внешнее соединение возвращает все строки одной или двух таблиц, которые участвуют в соединении.

Outer Join имеет следующий формальный синтаксис:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT столбцы  FROM таблица1  {LEFT|RIGHT} [OUTER] JOIN таблица2 ON условие1  [{LEFT|RIGHT} [OUTER] JOIN таблица3 ON условие2]... |

Перед оператором **JOIN** указывается одно из ключевых слов **LEFT** или **RIGHT**, которые определяют тип соединения:

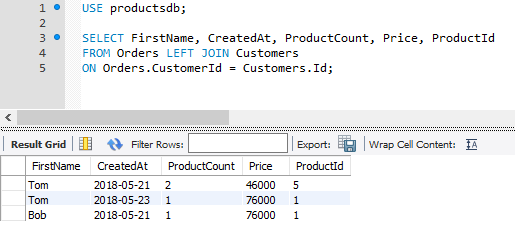
* **LEFT**: выборка будет содержать все строки из первой или левой таблицы
* **RIGHT**: выборка будет содержать все строки из второй или правой таблицы

Также перед оператором JOIN может указываться ключевое слово **OUTER**, но его применение необязательно. Далее после JOIN указывается присоединяемая таблица, а затем идет условие соединения.

Например, соединим таблицы Orders и Customers:

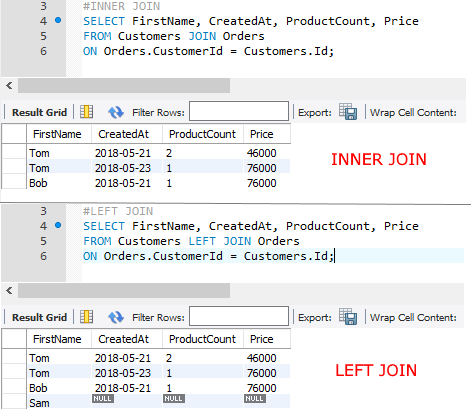
|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName, CreatedAt, ProductCount, Price, ProductId  FROM Orders LEFT JOIN Customers  ON Orders.CustomerId = Customers.Id |

Таблица Orders является первой или левой таблицей, а таблица Customers - правой таблицей. Поэтому, так как здесь используется выборка по левой таблице, то вначале будут выбираться все строки из Orders, а затем к ним по условию Orders.CustomerId = Customers.Id будут добавляться связанные строки из Customers.



По вышеприведенному результату может показаться, что левостороннее соединение аналогично INNER Join, но это не так. Inner Join объединяет строки из дух таблиц при соответствии условию. Если одна из таблиц содержит строки, которые не соответствуют этому условию, то данные строки не включаются в выходную выборку. Left Join выбирает все строки первой таблицы и затем присоединяет к ним строки правой таблицы. К примеру, возьмем таблицу Customers и добавим к покупателям информацию об их заказах:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9 | #INNER JOIN  SELECT FirstName, CreatedAt, ProductCount, Price  FROM Customers JOIN Orders  ON Orders.CustomerId = Customers.Id;    #LEFT JOIN  SELECT FirstName, CreatedAt, ProductCount, Price  FROM Customers LEFT JOIN Orders  ON Orders.CustomerId = Customers.Id; |

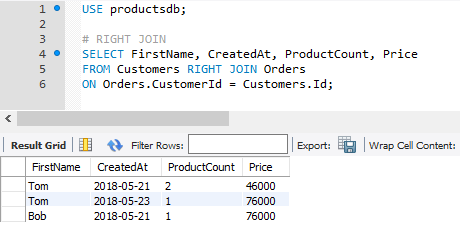


В случае с LEFT JOIN MySQL выбирает сначала всех покупателей из таблицы Customers, затем сопоставляет их с заказами из таблицы Orders через условие Orders.CustomerId = Customers.Id. Однако не у всех покупателей есть заказы. В этом случае покупателю для соответствующих столбцов устанавливаются значения NULL.

Изменим в примере выше тип соединения для OUTER JOIN с левостороннего на правостороннее:

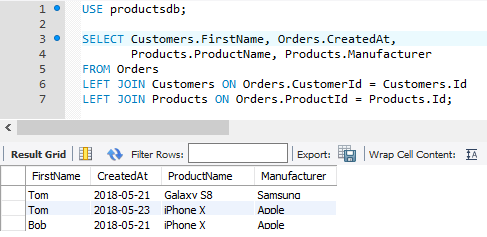
|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName, CreatedAt, ProductCount, Price  FROM Customers RIGHT JOIN Orders  ON Orders.CustomerId = Customers.Id; |

Теперь будут выбираться все строки из Orders (из правой таблицы), а к ним уже будет присоединяться связанные по условию строки из таблицы Customers:



Используем левостороннее соединение для добавления к заказам информации о пользователях и товарах:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT Customers.FirstName, Orders.CreatedAt,  Products.ProductName, Products.Manufacturer  FROM Orders  LEFT JOIN Customers ON Orders.CustomerId = Customers.Id  LEFT JOIN Products ON Orders.ProductId = Products.Id; |



И также можно применять более комплексные условия с фильтрацией и сортировкой. Например, выберем все заказы с информацией о клиентах и товарах по тем товарам, у которых цена меньше 45000, и отсортируем по дате заказа:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | SELECT Customers.FirstName, Orders.CreatedAt,  Products.ProductName, Products.Manufacturer  FROM Orders  LEFT JOIN Customers ON Orders.CustomerId = Customers.Id  LEFT JOIN Products ON Orders.ProductId = Products.Id  WHERE Products.Price > 45000  ORDER BY Orders.CreatedAt; |

Или выберем всех пользователей из Customers, у которых нет заказов в таблице Orders:

|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName FROM Customers  LEFT JOIN Orders ON Customers.Id = Orders.CustomerId  WHERE Orders.CustomerId IS NULL; |

Также можно комбинировать Inner Join и Outer Join:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | SELECT Customers.FirstName, Orders.CreatedAt,  Products.ProductName, Products.Manufacturer  FROM Orders  JOIN Products ON Orders.ProductId = Products.Id AND Products.Price > 45000  LEFT JOIN Customers ON Orders.CustomerId = Customers.Id  ORDER BY Orders.CreatedAt; |

Вначале по условию к таблице Orders через Inner Join присоединяется связанная информация из Products, затем через Outer Join добавляется информация из таблицы Customers.

## **UNION**

Оператор **UNION** позволяет обединить две однотипных выборки. Эти выборки могут быть из разных таблиц или из одной и той же таблицы. Формальный синтаксис объединения:

|  |  |
| --- | --- |
| 1  2  3 | SELECT\_выражение1  UNION [ALL] SELECT\_выражение2  [UNION [ALL] SELECT\_выражениеN] |

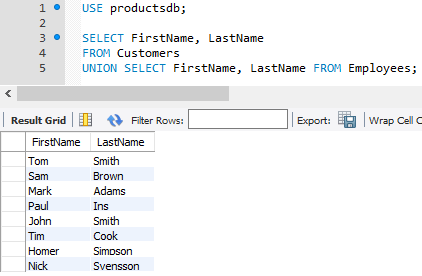
Например, пусть в базе данных будут две отдельные таблицы для клиентов банка (таблица Customers) и для сотрудников банка (таблица Employees):

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29 | CREATE TABLE Customers  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  FirstName VARCHAR(20) NOT NULL,  LastName VARCHAR(20) NOT NULL,  AccountSum DECIMAL  );  CREATE TABLE Employees  (  Id INT AUTO\_INCREMENT PRIMARY KEY,  FirstName VARCHAR(20) NOT NULL,  LastName VARCHAR(20) NOT NULL  );    INSERT INTO Customers(FirstName, LastName, AccountSum)  VALUES  ('Tom', 'Smith', 2000),  ('Sam', 'Brown', 3000),  ('Mark', 'Adams', 2500),  ('Paul', 'Ins', 4200),  ('John', 'Smith', 2800),  ('Tim', 'Cook', 2800);    INSERT INTO Employees(FirstName, LastName)  VALUES  ('Homer', 'Simpson'),  ('Tom', 'Smith'),  ('Mark', 'Adams'),  ('Nick', 'Svensson'); |

Здесь мы можем заметить, что обе таблицы, несмотря на наличие различных данных, могут характеризоваться двумя общими атрибутами - именем (FirstName) и фамилией (LastName). Выберем сразу всех клиентов банка и его сотрудников из обеих таблиц:

|  |  |
| --- | --- |
| 1  2  3 | SELECT FirstName, LastName  FROM Customers  UNION SELECT FirstName, LastName FROM Employees; |

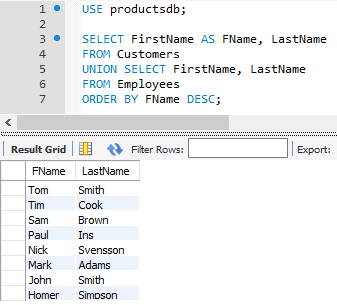
Здесь из первой таблицы выбираются два значения - имя и фамилия клиента. Из второй таблицы Employees также выбираются два значения - имя и фамилия сотрудников. То есть при объединении количество выбираемых столбцов и их тип совпадают для обеих выборок.



При этом названия столбцов объединенной выборки будут совпадать с названия столбцов первой выборки. И если мы захотим при этом еще произвести сортировку, то в выражениях **ORDER BY** необходимо ориентироваться именно на названия столбцов первой выборки:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT FirstName AS FName, LastName  FROM Customers  UNION SELECT FirstName, LastName  FROM Employees  ORDER BY FName DESC; |

В данном случае каждая выборка имеет по столбцу FName из первой выборки. Тем не менее при сортировке будет учитываться и значение столбца FirstName из второй выборки:

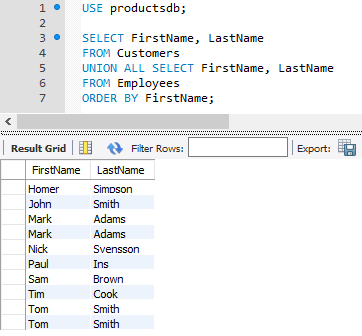


Если же в одной выборке больше столбцов, чем в другой, то они не смогут быть объединены. Например, в следующем случае объединение завершится с ошибкой:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT FirstName, LastName, AccountSum  FROM Customers  UNION SELECT FirstName, LastName  FROM Employees; |

Если оба объединяемых набора содержат в строках идентичные значения, то при объединении повторяющиеся строки удаляются. Например, в случае с таблицами Customers и Employees сотрудники банка могут быть одновременно его клиентами и содержаться в обеих таблицах. При объединении в примерах выше всех дублирующиеся строки удалялись. Если же необходимо при объединении сохранить все, в том числе повторяющиеся строки, то для этого необходимо использовать оператор **ALL**:

|  |  |
| --- | --- |
| 1  2  3  4  5 | SELECT FirstName, LastName  FROM Customers  UNION ALL SELECT FirstName, LastName  FROM Employees  ORDER BY FirstName; |



Объединять выборки можно и из одной и той же таблицы. Например, в зависимости от суммы на счете клиента нам надо начислять ему определенные проценты:

|  |  |
| --- | --- |
| 1  2  3  4 | SELECT FirstName, LastName, AccountSum + AccountSum \* 0.1 AS TotalSum  FROM Customers WHERE AccountSum < 3000  UNION SELECT FirstName, LastName, AccountSum + AccountSum \* 0.3 AS TotalSum  FROM Customers WHERE AccountSum >= 3000; |

В данном случае если сумма меньше 3000, то начисляются проценты в размере 10% от суммы на счете. Если на счете больше 3000, то проценты увеличиваются до 30%.

