

# MYSQL. JOIN

# ТЕОРИЯ

- JOIN, в переводе на великий и могучий, означает "объединять", то есть собирать из нескольких кусочков единое целое. В базе данных MySQL такими "кусочками" служат столбцы таблиц, которые можно объединять при выборке.
- Объединения позволяют извлекать данные из нескольких таблиц без создания временных таблиц и за один запрос.

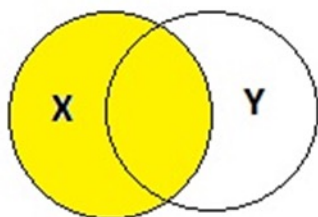
# ТЕОРИЯ

В зависимости от требований к результату, MySQL позволяет производить три разных типа объединения:

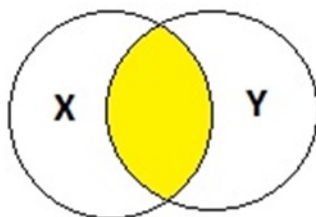
- ◎ INNER JOIN
- ◎ OUTER JOIN
- ◎ CROSS JOIN

В свою очередь, OUTER JOIN может быть LEFT и RIGHT (слово OUTER часто опускается).

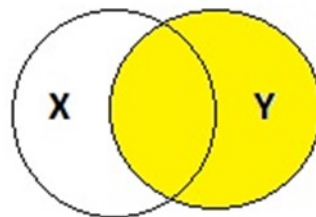
# SQL JOINS



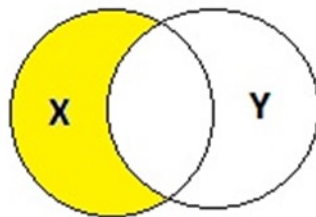
Select X.\*,Y.\*  
From X Left Join Y  
On X.key = Y.key



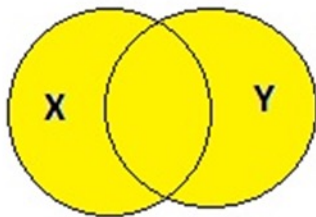
Select X.\*,Y.\*  
From X Inner Join Y  
On X.key = Y.key



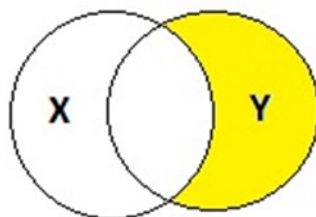
Select X.\*,Y.\*  
From X Right Join Y  
On X.key = Y.key



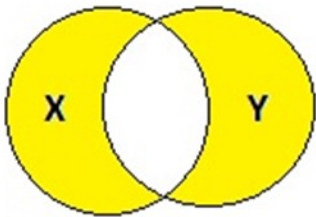
Select X.\*  
From X Left Join Y  
On X.key = Y.key  
Where Y.key is null



Select X.\*,Y.\*  
From X Full Outer Join Y  
On X.key = Y.key



Select Y.\*  
From X Right Join Y  
On X.key = Y.key  
Where X is Null



Select X.\*,Y.\*  
From X Full Outer Join Y  
On X.key = Y.key  
Where X.key is Null or Y.key is Null

# ТАБЛИЦЫ "ТОВАРЫ" И "ОПИСАНИЯ"

- Таблица с наименованием товаров хранит номер товара (id) и краткое название (name) и таблица с описанием товаров

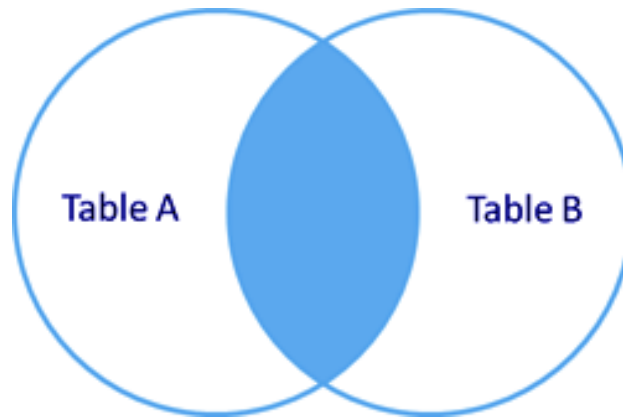
id	name
1	Книга
2	Табуретка
3	Карандаш

id	description
1	Замечательная книга
3	Красный карандаш
5	Зелёная машинка

Таблица `nomenclature` содержит перечень всех товаров, которые есть в базе. Таблица описаний `description`, напротив, содержит лишь неполный перечень описаний для товаров, которые необязательно присутствуют в базе. Чтобы однозначно привязать описание к товару, в таблицах присутствует столбец `id`, который содержит уникальный номер товара.

## INNER JOIN (CROSS JOIN) - ВНУТРЕННЕЕ (ПЕРЕКРЁСТНОЕ) ОБЪЕДИНЕНИЕ

- Этот тип объединения позволяет извлекать строки, которые обязательно присутствуют во всех объединяемых таблицах.





# INNER JOIN

В простейшем случае (без указания условий отбора), выборка вернёт т.н. декартово произведение, в котором каждая строка одной таблицы будет сопоставлена с каждой строкой другой таблицы:

**SELECT \* FROM nomenclature INNER JOIN description;**

id	name	id	description
1	Книга	1	Замечательная книга
2	Табуретка	1	Замечательная книга
3	Карандаш	1	Замечательная книга
1	Книга	3	Красный карандаш
2	Табуретка	3	Красный карандаш
3	Карандаш	3	Красный карандаш
1	Книга	5	Зелёная машинка
2	Табуретка	5	Зелёная машинка
3	Карандаш	5	Зелёная машинка

## INNER JOIN (АЛЬТЕРНАТИВНЫЕ ВАРИАНТЫ)

Помимо конструкции INNER JOIN внутреннее объединение можно объявить так же через CROSS JOIN, JOIN и запятую в объявлении FROM. Следующие четыре запроса вернут одинаковый результат:

- ◎ **SELECT \* FROM nomenclature INNER JOIN description;**
- ◎ **SELECT \* FROM nomenclature CROSS JOIN description;**
- ◎ **SELECT \* FROM nomenclature JOIN description;**
- ◎ **SELECT \* FROM nomenclature, description;**



# INNER JOIN

Как правило, декартово произведение таблиц требуется нечасто, чаще требуется выбрать только те записи, которые сопоставлены друг другу. Сделать это можно, если задать условие отбора, используя ON или USING.

```
SELECT * FROM nomenclature INNER JOIN description using(id);
```

id	name	description
1	Книга	Замечательная книга
3	Карандаш	Красный карандаш

Запрос вернул только две записи, поскольку именно столько строк имеют одинаковые идентификаторы в обеих таблицах.

Использование USING обусловлено тем, что в таблицах ключевой столбец имеет одно и тоже имя - id. В противном случае, надо было бы использовать ON.

```
SELECT * FROM nomenclature INNER JOIN description ON  
nomenclature.id = description.id;
```

# INNER JOIN (',')

Если объединять таблицы через запятую, то нельзя использовать конструкции ON и USING, поэтому условие может быть задано только в конструкции WHERE.

Например, это может выглядеть так:

```
SELECT * FROM nomenclature, description WHERE  
nomenclature.id = description.id;
```

id	name	id	description
1	Книга	1	Замечательная книга
3	Карандаш	3	Красный карандаш

# INNER JOIN (2 СПОСОБА)

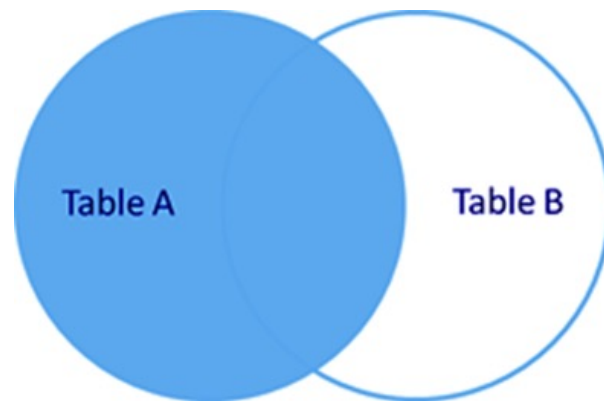
Код - способы объявления внутреннего объединения таблиц

```
SELECT * FROM Таблица1, Таблица2[,  
Таблица3, ...] [WHERE Условие1 [Условие2  
...]
```

```
SELECT * FROM Таблица1 [INNER | CROSS]  
JOIN Таблица2 [(ON Условие1 [Условие2  
...]) | (USING(Поле))]
```

# LEFT JOIN

Левосторонние объединения позволяют извлекать данные из таблицы, дополняя их по возможности данными из другой таблицы.



# LEFT JOIN

К примеру, чтобы получить полный список наименований товаров вместе с их описанием, нужно выполнить следующий запрос:

```
SELECT * FROM nomenclature LEFT JOIN description  
USING(id);
```

id	name	description
1	Книга	Замечательная книга
2	Табуретка	NULL
3	Карандаш	Красный карандаш

Поскольку для наименования Табуретка в таблице описаний нет подходящей записи, то в поле description подставился NULL. Это справедливо для всех записей, у которых нет подходящей пары.

# LEFT JOIN

Если дополнить предыдущий запрос условием на проверку несуществования описания, то можно получить список записей, которые не имеют пары в таблице описаний:

```
SELECT id, name FROM nomenclature LEFT JOIN  
description USING(id) WHERE description IS NULL;
```

id	name
2	Табуретка

- По сути это и есть основное назначение внешних запросов - показывать расхождение данных двух таблиц.
- Кроме того, при таком объединении обязательным является условие, которое задаётся через ON или USING. Без него запрос будет выдавать ошибку.

# RIGHT JOIN

Этот вид объединений практически ничем не отличается от левостороннего объединения, за тем исключением, что данные берутся из второй таблицы, которая находится справа от конструкции JOIN, и сравниваются с данными, которые находятся в таблице, указанной перед конструкцией.

**SELECT \* FROM nomenclature RIGHT JOIN description USING(id);**

id	description	name
1	Замечательная книга	Книга
3	Красный карандаш	Карандаш
5	Зелёная машинка	NULL

Как видно, теперь уже поле name содержит нулевые значения. Также поменялся и порядок расположения столбцов.

Однако, во всех случаях использования правосторонних объединений, запрос можно переписать, используя левостороннее объединение, просто поменяв таблицы местами, и наоборот. Следующие два запроса равнозначны:

**SELECT \* FROM nomenclature LEFT JOIN description USING(id);**  
**SELECT \* FROM description RIGHT JOIN nomenclature USING(id);**



# CROSS JOIN

Тип CROSS JOIN применяется если необходимо получить все возможные сочетания из обеих таблиц. Condition для этого типа оператора JOIN не указывается.

```
◎ SELECT t_resources.t_name,  
       t_users.t_nick FROM t_resources CROSS  
       JOIN t_users
```

Этот вид объединения следует использовать с большой осторожностью, поскольку он снижает производительность и часто ( что кстати видно из примера) содержит избыточную информацию.

# МНОГОТАБЛИЧНЫЕ ЗАПРОСЫ

Используя JOIN, можно объединять не только две таблицы, но и гораздо больше.

Помимо объединений разных таблиц, MySQL позволяет объединять таблицу саму с собой. Однако, в любом случае необходимо следить за именами столбцов и таблиц, если они будут неоднозначны, то запрос не будет выполнен.

Так, если таблицу просто объединить саму на себя, то возникнет конфликт имён и запрос не выполнится.

```
SELECT * FROM nomenclature JOIN nomenclature;
```

```
ERROR 1066 (42000): Not unique table/alias:  
'nomenclature'
```

# МНОГОТАБЛИЧНЫЕ ЗАПРОСЫ (SELECT)

Обойти конфликт имён позволяет использование синонимов (alias) для имён таблиц и столбцов. В следующем примере внутреннее объединение будет работать успешнее:

```
SELECT * FROM nomenclature AS t1 JOIN nomenclature AS  
t2 LEFT JOIN nomenclature AS t3 ON t1.id = t3.id AND  
t2.id = t1.id;
```

id	name	id	name	id	name
1	Книга	1	Книга	1	Книга
2	Табуретка	1	Книга	NULL	NULL
3	Карандаш	1	Книга	NULL	NULL
1	Книга	2	Табуретка	NULL	NULL
2	Табуретка	2	Табуретка	2	Табуретка
3	Карандаш	2	Табуретка	NULL	NULL
1	Книга	3	Карандаш	NULL	NULL
2	Табуретка	3	Карандаш	NULL	NULL
3	Карандаш	3	Карандаш	3	Карандаш

# МНОГОТАБЛИЧНЫЕ ЗАПРОСЫ (UPDATE И DELETE)

Помимо выборок использовать объединения можно также и в запросах UPDATE и DELETE

Так, следующие три запроса проделывают одинаковую работу:

- 1) **UPDATE nomenclature AS t1, nomenclature AS t2 SET t1.id = t2.id WHERE t1.id = t2.id;**
- 2) **UPDATE nomenclature AS t1 JOIN nomenclature AS t2 SET t1.id = t2.id WHERE t1.id = t2.id;**
- 3) **UPDATE nomenclature AS t1 JOIN nomenclature AS t2 USING(id) SET t1.id = t2.id;**

## МНОГОТАБЛИЧНЫЕ ЗАПРОСЫ (DELETE)

```
DELETE t1 FROM nomenclature AS t1 JOIN  
nomenclature AS t2 USING(id) WHERE t2.id >  
10;
```

Следует помнить, что при использовании многотабличных запросов на удаление или обновление данных, нельзя включать в запрос конструкции ORDER BY и LIMIT. Впрочем, это ограничение очень эффективно обходится при помощи временных таблиц, просто, надо это учитывать при модификации однотабличных запросов.

# ПРИМЕРЫ С ГРУППИРОВКОЙ

По каждому поставщику вычисляются сумма и количество поступивших от него товаров:

```
SELECT supplier_id, product_id, SUM(amount) AS amount_sum,  
SUM(amount*price) AS income_sum  
FROM m_income AS a INNER JOIN m_product AS b ON  
a.product_id=b.id  
GROUP BY supplier_id, product_id;
```

Для каждого поставщика вычисляются сумма и количество его продуктов, проданных нами:

```
SELECT supplier_id, product_id, SUM(amount) AS amount_sum,  
SUM(amount*price) AS outcome_sum  
FROM m_outcome AS a INNER JOIN m_product AS b ON  
a.product_id=b.id  
GROUP BY supplier_id, product_id;
```