

Базы данных

А5. Анализ данных (расширение)



Московский государственный технический университет
имени Н.Э. Баумана

Факультет ИБМ

Апр 2025 года

Москва

Артемьев Валерий Иванович © 2025

Расширение анализа данных

Соединение таблиц JOIN

И
Д

Многомерный анализ данных

О
П
И
С
Р
С

Соединение таблиц JOIN



Инструкции JOIN соединяют две и более *таблицы по согласованным связям* , являются основой много-табличных представлений (запросов),обеспечивают добавление столбцов, расшифровку справочных кодов, очистку и обогащение данных.

SELECT *список полей из одной или нескольких таблиц*

FROM *таблица1*

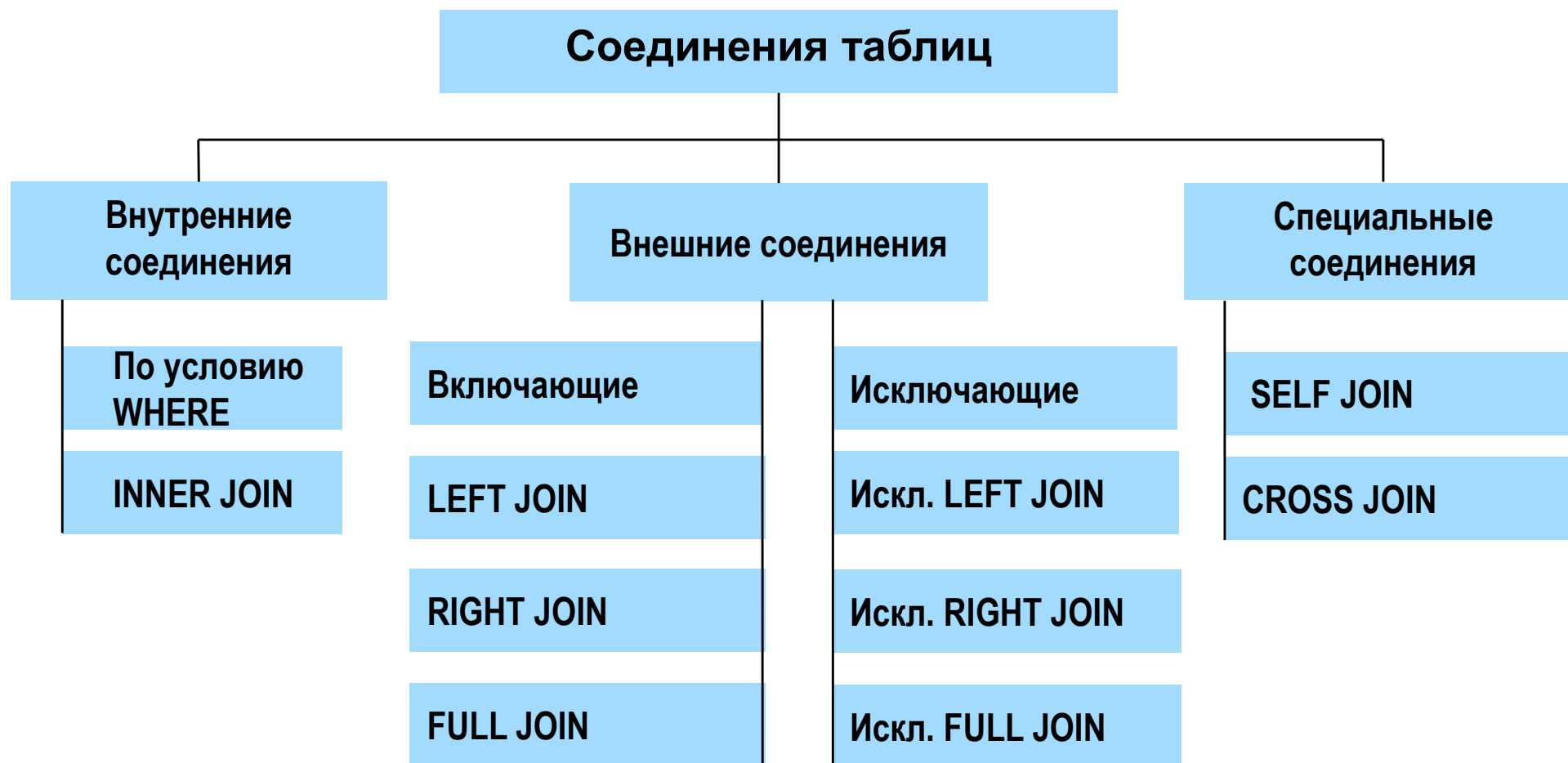
Операция **JOIN** *таблица2* **ON** *условие соединения 1*

...

WHERE *общие условия отбора записей*

ORDER BY *порядок сортировки*

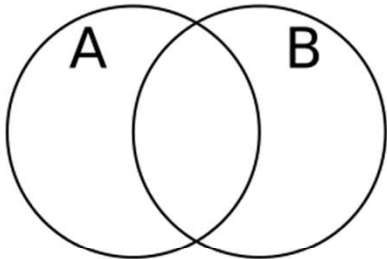
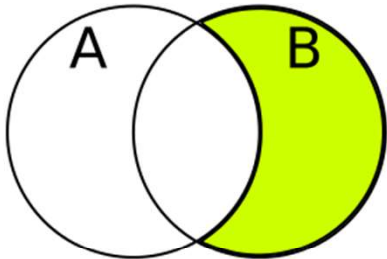
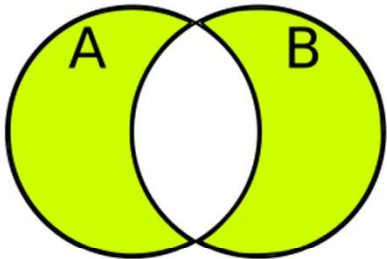
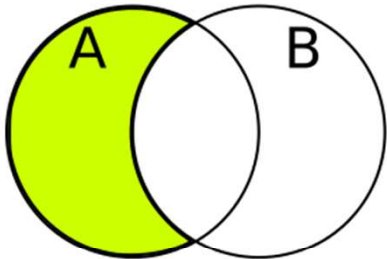
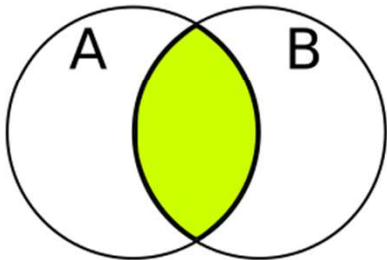
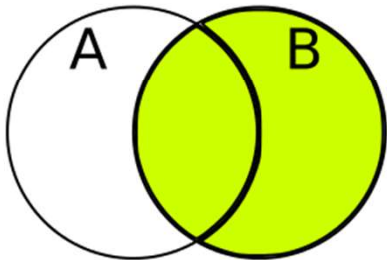
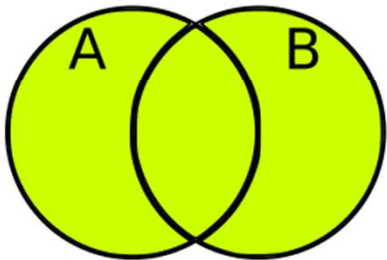
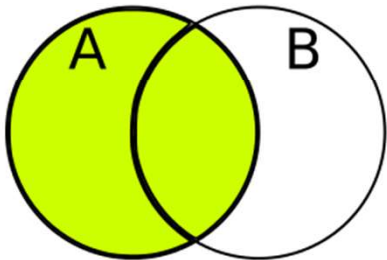
Классификация соединений таблиц JOIN



Применение разных видов соединений

- Когда результат должен содержать только данные двух таблиц с общим ключом, применяют INNER JOIN или просто JOIN.
- Если нужен полный список записей одной из таблиц, объединенных с данными из другой, используют операторы LEFT или содержать полный список записей обеих таблиц, где некоторые записи объединены, применяют оператор FULL JOIN.
- Если нужно декартово произведение двух таблиц, используют оператор CROSS JOIN.
- Хотите соединить данные из одной и той же таблицы между собой — нужен SELF JOIN.

Диаграммы Венна для соединений

No join (\emptyset)	[Exclusive] Right Join ($\neg A$)	[Exclusive] Full Join ($A \oplus B$)	[Exclusive] Left Join ($\neg B$)
	 <pre>SELECT * FROM A RIGHT JOIN B ON A.key = B.key WHERE B.key IS NULL</pre>	 <pre>SELECT * FROM A FULL JOIN B ON A.key = B.key WHERE B.key IS NULL OR A.key IS NULL</pre>	 <pre>SELECT * FROM A LEFT JOIN B ON A.key = B.key WHERE B.key IS NULL</pre>
Inner Join ($A \cap B$)	[Inclusive] Right Join (B)	[Inclusive] Full Join ($A \cup B$)	[Inclusive] Left Join (A)
 <pre>SELECT * FROM A INNER JOIN B ON A.key = B.key</pre>	 <pre>SELECT * FROM A RIGHT JOIN B ON A.key = B.key</pre>	 <pre>SELECT * FROM A FULL JOIN B ON A.key = B.key</pre>	 <pre>SELECT * FROM A LEFT JOIN B ON A.key = B.key</pre>

Тестовая БД для JOIN



Persons (Сотрудники)

id_person	name	position_ref
1	Владимир	1
2	Татьяна	2
3	Александр	6
4	Борис	2

Столбец *position_ref*
это ссылка на
справочник должностей

Positions (должности)

id_pos	title
1	Дизайнер
2	Редактор
3	Программист

Справочник должностей

Внутреннее соединение INNER JOIN



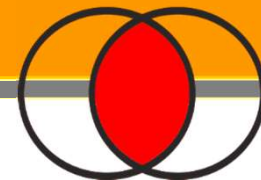
```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
INNER JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref;
```

id_person	name	id_pos	title
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор

Такое соединение покажет нам данные из таблиц, только если условие связывания соблюдается — т.е. для сотрудника указан существующий в справочнике идентификатор должности.

Если поменять порядок соединения таблиц — получим тот же результат.

Внутреннее соединение по условию WHERE

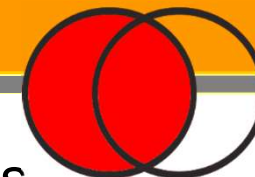


```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons, Positions  
WHERE Persons.id_posid_pos = Positions.position_ref;
```

id_person	name	id_pos	title
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор

Эквивалентно INNER JOIN, даёт аналогичный результат.

Внешнее левое соединение LEFT JOIN



```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
LEFT JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref;
```

id_person	name	id_pos	title
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор
3	Александр	NULL	NULL

Внешнее соединение включает в себя результаты запроса INNER и добавляются «неиспользованные» строки из правой таблицы.

Левое подмножество соединения

Exclusive LEFT JOIN



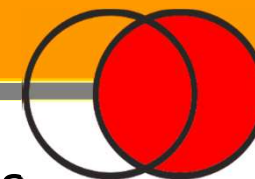
```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
LEFT JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref  
WHERE Persons.id_pos is NULL;
```

id_person	name	id_pos	title
3	Александр	NULL	NULL

Этот пример показывает, как исключить пересечение и получить только левую часть множества.

LEFT JOIN ограничиваем проверкой, что данных из 2-ой таблицы нет. Получим тех персон, у которых нет должности с указанным ключом.

Внешнее правое соединение RIGHT JOIN



```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
FULL JOIN Positions  
ON Persons.id_posid_pos = Positions.position_ref;
```

id_person	name	id_pos	title
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор
NULL	NULL	3	Программист

Тут данные из левой таблицы присоединяются к правой. Справочник должностей (правая таблица) содержит «неиспользуемую» запись с id_pos=3 — «программист». Теперь она попала в результат запроса.

Правое подмножество соединения Exclusive RIGHT JOIN



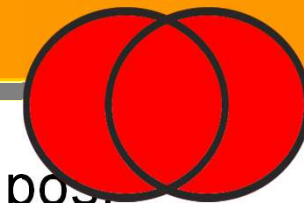
```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
RIGHT JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref  
WHERE Persons.id_person is NULL;
```

id_person	name	id_pos	title
NULL	NULL	3	Программист

Пример показывает, как исключить пересечение и получить только правую часть множества RIGHT JOIN, добавив проверку, что данных из 1-ой таблицы нет.

В результате получим никем не занятые должности.

Внешнее полное соединение FULL JOIN

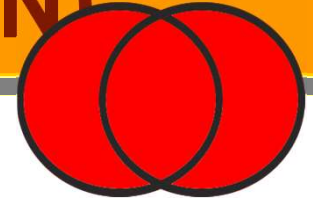


```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
FULL JOIN Positions  
ON Persons.id_posid_pos = Positions.position_ref;
```

id_person	name	id_pos	title
1	Владимир	1	Дизайнер
2	Татьяна	2	Редактор
4	Борис	2	Редактор
3	Александр	NULL	NULL
NULL	NULL	3	Программист

В результате такого соединения получим все записи из левой и правой таблицы.

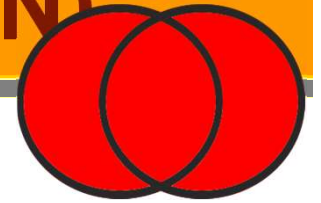
1-й способ имитации полного соединения (LEFT JOIN и RIGHT JOIN)



```
(SELECT    Persons.id_person, Persons.name, Persons.id_pos,  
           Positions.title  
FROM      Persons  
LEFT JOIN Positions  
           ON Persons.id_posid_pos = Positions.position_ref)  
UNION  
(SELECT    Persons.id_person, Persons.name, Persons.id_pos,  
           Positions.title  
FROM      Persons  
RIGHT JOIN Positions  
           ON Persons.id_posid_pos = Positions.position_ref);
```

При таком вызове UNION, после слияния результатов будут отсечены дубли (как при DISTINCT). Для отсечения нужна сортировка, что может сказываться на быстродействии.

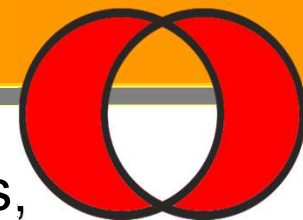
2-й способ имитации полного соединения (LEFT JOIN и RIGHT JOIN)



```
(SELECT    Persons.id_person, Persons.name, Persons.id_pos,  
           Positions.title  
FROM      Persons  
LEFT JOIN Positions  
           ON Persons.id_posid_pos = Positions.position_ref)  
UNION ALL  
(SELECT    Persons.id_person, Persons.name, Persons.id_pos,  
           Positions.title  
FROM      Persons  
RIGHT JOIN Positions  
           ON Persons.id_posid_pos = Positions.position_ref  
WHERE      Persons.id_person is NULL);
```

Объединение LEFT и RIGHT, но в одном из запросов мы исключаем часть, соответствующую INNER. Используем UNION ALL, что позволяет обойтись без сортировки.

Исключающее полное соединение Exclusive FULL JOIN

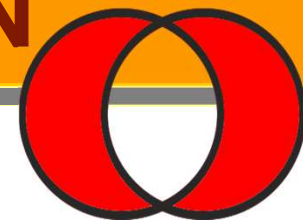


```
SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title FROM Persons  
FULL JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref  
WHERE Persons.id_pos is NULL  
        OR Persons.id_person is NULL
```

id_person	name	id_pos	title
3	Александр	NULL	NULL
NULL	NULL	3	Программист

Всё, кроме пересечения — этот запрос соберет все случаи, когда по какой-то причине данные из таблиц не связаны.

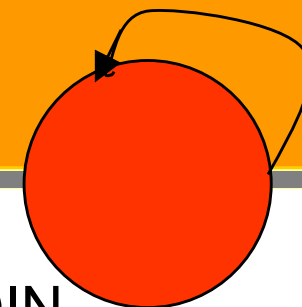
Имитация исключающего FULL JOIN



```
(SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
LEFT JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref  
WHERE Persons.id_pos is NULL)  
UNION ALL  
(SELECT Persons.id_person, Persons.name, Persons.id_pos,  
        Positions.title  
FROM Persons  
RIGHT JOIN Positions  
        ON Persons.id_posid_pos = Positions.position_ref  
WHERE Persons.id_person is NULL)
```

Исключающее пересечение множеств FULL JOIN можно получить, сложив левое и правое подмножества соединений через UNION ALL (т.к. подмножества не пересекаются).

Соединение SELF JOIN



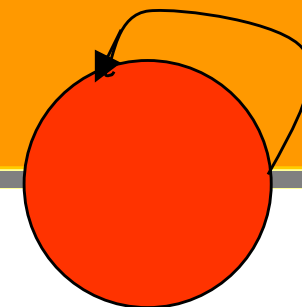
Выполняется соединение данных из одной и той же таблицы. В SQL нет отдельного оператора для SELF JOIN соединения, поэтому пользуйтесь операторами разными видами JOIN или WHERE. Чтобы различать оригинальную таблицу и её копию, применяйте алиасы

```
SELECT A.column1, B.column2  
FROM table_name AS A  
JOIN table_name AS B  
ON A.common_field = B.common_field;
```

Используется для решения различных аналитических задач:

- нужно сравнить строки в одной и той же таблице
- извлечь данные из разных строк, которые связаны между собой.
- объединение записей внутри одной таблицы.
- развёртка оргструктуры: найти сотрудников каждого руководителя,
- анализ временных рядов данных.

Пример SELF JOIN: Менеджеры и их подчинённые



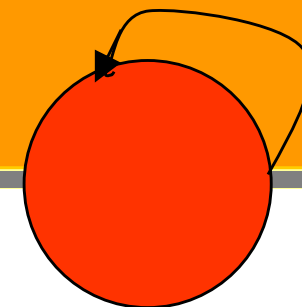
Имеется таблица о сотрудниках **Employees** с указанием ID менеджера. Нужно найти всех сотрудников, которые работают под одним и тем же менеджером.

Соединим таблицу **Employees** с самой собой, и затем отфильтруем результаты, чтобы исключить совпадения одного и того же сотрудника.

```
SELECT A.employee_name AS Сотрудник1,  
       B.employee_name AS Сотрудник2  
FROM Employees AS A  
JOIN Employees AS B  
ON A.manager_id = B.manager_id  
WHERE A.employee_id <> B.employee_id;
```

Получим пары сотрудников, работающих под одним и тем же менеджером.

Пример SELF JOIN: Иерархические структуры



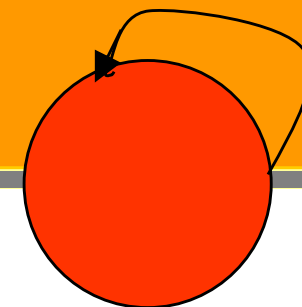
SELF JOIN можно использовать для работы с иерархиями данных, такими как оргструктуры или каталоги продуктов. Пусть имеется таблица **Categories** с информацией о категориях и подкатегориях продуктов, можно применить SELF JOIN для извлечения всех подкатегорий для каждой категории.

```
SELECT A.category_name AS Parent_Category,  
       B.category_name AS Sub_Category  
FROM Categories AS A  
JOIN Categories AS B  
ON A.category_id = B.parent_category_id;
```

Это позволяет нам анализировать иерархические структуры данных и выявлять взаимосвязи между различными уровнями иерархии.

Пример SELF JOIN:

Нахождение предыдущей записи



Имеется таблица **Orders** с информацией о заказах, включая ID заказа и дату заказа. Нужно найти предыдущий заказ для каждого заказа. Это полезно для анализа последовательности заказов и выявления тенденций в поведении клиентов.

Используем SELF JOIN, чтобы соединить таблицу Orders с самой собой, и находим предыдущий заказ для каждого заказа на основе даты заказа.

```
SELECT A.order_id, A.order_date,  
       B.order_id AS Previous_Order_ID,  
       B.order_date AS Previous_Order_Date  
FROM Orders AS A  
LEFT JOIN Orders AS B  
  ON A.order_date > B.order_date  
WHERE B.order_date IS NOT NULL  
ORDER BY A.order_date;
```

Это позволяет нам анализировать временные ряды данных и выявлять

Перекрёстное соединение CROSS JOIN

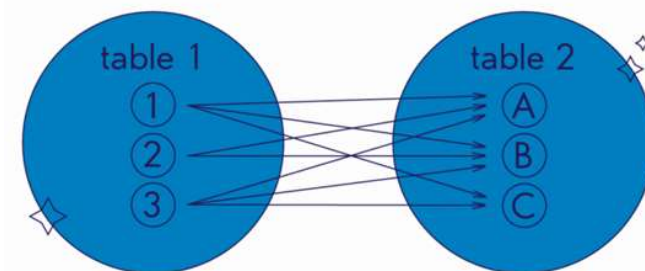
Оператор соединения, создающий **декартово произведение** двух таблиц. Прекрасно подходит для генерации всех возможных **комбинаций** данных из этих таблиц.

Порождает **большое число записей**, равное произведению количества записей соединяемых таблиц.

Чтобы исключить "Чернобыль данных", нужно использовать **механизмы фильтрации**.

Используется:

- при разработке **характеристик продуктов**
- для составления **расписаний игровых турниров**
- для создания **многоязычных словарей**
- при разработке подробного **плана техобслуживания оборудования**, комбинируя его с требуемыми проверками и процедурами.
- для генерации **справочника календаря**
- для генерации **тестовых данных**, покрывающих все возможные сценарии, что помогает выявлять дефекты.



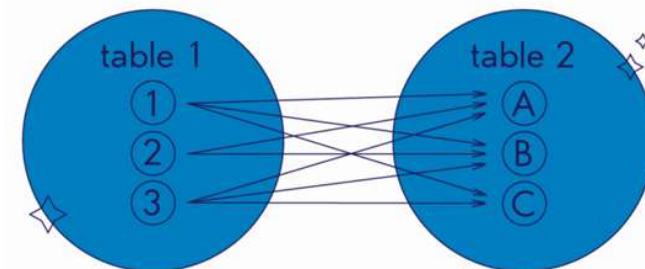
Пример CROSS JOIN: генерация характеристик продукта

Для примера сгенерируем сочетания цветов и размеров футболок, соединив перекрёстно таблицы цветов и размеров футболок:

```
SELECT Shirt_Colors.Color, Shirt_Sizes.Size  
FROM Shirt_Colors  
CROSS JOIN Shirt_Sizes;
```

Или аналогичный вариант:

```
SELECT Shirt_Colors.Color, Shirt_Sizes.Size  
FROM Shirt_Colors, Shirt_Sizes;
```



Shirt_Colors.Color	Shirt_Sizes.Size

**Терпения и удачи всем,
кто связан с базами данных**

Спасибо за внимание!

Валерий Иванович Артемьев

МГТУ имени Н.Э. Баумана, кафедра ИУ-5

Банк России

Департамент данных, проектов и процессов

Тел.: +7(495) 753-96-25

e-mail: viart@bmstu.ru