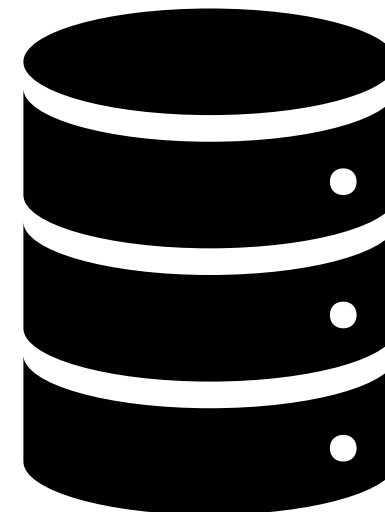


# Базы данных

Лекция 3. СУБД. Троичная логика. SQL: продолжение.



ФПМИ МФТИ,  
2022

# СИСТЕМА УПРАВЛЕНИЯ БАЗАМИ ДАННЫХ

- совокупность программных и лингвистических средств, обеспечивающих управление созданием и использованием баз данных

# ПРЕИМУЩЕСТВА

- Совместный доступ
  - Экономия на обслуживании
- Непротиворечивость
  - Единая версия истины
- Контроль целостности
  - Соответствие данных внутренней логике системы
- Транзакционность
  - Надежный механизм обновления
- Независимость от данных
  - Упрощение сопровождения

# ОСНОВНЫЕ СУЩЕСТВУЮЩИЕ РСУБД

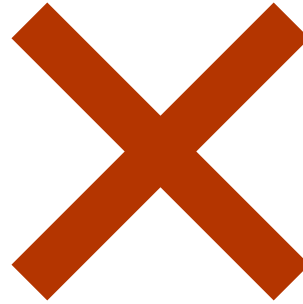


- PostgreSQL
  - Open source and free
  - Наиболее соответствует стандарту SQL
- Oracle
  - Первыми создали коммерческое решение (1979 г.)
  - Много дополнительных плюшек
- MS SQL
  - Оригинальная идея получена от Sysbase
  - Хорошая интеграция с Microsoft решениями
  - Разнообразие процедурных расширений
- MySQL
  - Тоже open source
  - Тоже производится Oracle

# БИНАРНАЯ ЛОГИКА



TRUE



FALSE

# БИНАРНАЯ ЛОГИКА

<u>AND</u>	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

<u>OR</u>	TRUE	FALSE
TRUE	TRUE	TRUE
FALSE	TRUE	FALSE

	<u>NOT</u>
TRUE	FALSE
FALSE	TRUE

# БИНАРНАЯ ЛОГИКА



TRUE



FALSE

АБСТРАГИРУЕМСЯ ОТ ПРИВЫЧНОГО ИЛИ ОЧЕНЬ СИЛЬНО ГРУСТИМ

# ТЕРНАРНАЯ (ТРОИЧНАЯ) ЛОГИКА

- Что делать, если значение в ячейке неизвестно?
- Как проверить, что значение действительно неизвестно?
- Как учитывать в расчетах такие ячейки?
- Как правильно их агрегировать?



# ТЕРНАРНАЯ (ТРОИЧНАЯ) ЛОГИКА

---

- Допустимые значения логического выражения:
  - TRUE
  - FALSE
  - UNKNOWN
- Как оперировать со значениями только из булевой логики
- Как оперировать со значениями типа UNKNOWN
- В SQL используется обозначение NULL



# СРАВНЕНИЕ С NULL

- `NULL = 1`
- `NULL <> 1`
- `NULL > 1`
- `NULL = NULL`



# СРАВНЕНИЕ С NULL

- `NULL = 1`            **NULL**
- `NULL <> 1`           **NULL**
- `NULL > 1`            **NULL**
- `NULL = NULL`        **NULL**



# NULL В ВЫРАЖЕНИЯХ С AND/OR

- (NULL = 1) OR (1 = 0)
- (NULL = 1) OR (1 = 1)
- (NULL = 1) AND (1 = 0)
- (NULL = 1) AND (1 = 1)
- NOT (NULL = NULL)



# NULL В ВЫРАЖЕНИЯХ С AND/OR

- `(NULL = 1) OR (1 = 0)`
  - `NULL OR FALSE`
- `(NULL = 1) OR (1 = 1)`
  - `NULL OR TRUE`
- `(NULL = 1) AND (1 = 0)`
  - `NULL AND FALSE`
- `(NULL = 1) AND (1 = 1)`
  - `NULL AND TRUE`
- `NOT (NULL = NULL)`
  - `NOT NULL`



# NULL В ВЫРАЖЕНИЯХ С AND/OR

- X AND TRUE
- X AND FALSE
- X OR TRUE
- X OR FALSE



# NULL В ВЫРАЖЕНИЯХ С AND/OR

- |               |       |
|---------------|-------|
| • X AND TRUE  | X     |
| • X AND FALSE | FALSE |
| • X OR TRUE   | TRUE  |
| • X OR FALSE  | X     |



# NULL В ВЫРАЖЕНИЯХ С AND/OR

• NULL OR FALSE	NULL
• NULL OR TRUE	TRUE
• NULL AND FALSE	FALSE
• NULL AND TRUE	NULL
• NOT NULL	NULL



Т.е. если NULL как-то влияет на значение логического выражения, результат не определен



# NULL В ЗАПРОСАХ

- Ключевые слова

- WHERE
- HAVING

строго требуют значение TRUE, поэтому условие NOT FALSE не является достаточным

# NULL В ЗАПРОСАХ

```
SELECT col  
FROM T  
WHERE col = NULL;
```

# NULL В ЗАПРОСАХ

```
SELECT col  
  FROM T  
 WHERE col = NULL;
```

По такому запросу не будет отображена ни одна запись

# NULL В ЗАПРОСАХ

```
SELECT col  
  FROM T  
 WHERE col = NULL  
        OR NOT (col = NULL) ;
```

# NULL В ЗАПРОСАХ

```
SELECT col  
  FROM T  
 WHERE col = NULL  
        OR NOT (col = NULL) ;
```

По такому запросу тоже не будет отображена ни одна запись

# NULL В ЗАПРОСАХ

- Предикат `IS NULL` позволяет отбирать строки со значениями `NULL`

```
SELECT col  
  FROM T  
 WHERE col IS NULL;
```

# NULL В ЗАПРОСАХ

- Предикат `IS NULL` позволяет отбирать строки со значениями `NULL`

```
SELECT col  
  FROM T  
 WHERE col IS NULL;
```

# NULL В ЗАПРОСАХ

- 1 NOT IN (0)
- 1 NOT IN (1)



# NULL В ЗАПРОСАХ

- 1 NOT IN (0)
- 1 NOT IN (1)
- 1 NOT IN (NULL)
- 1 NOT IN (2, NULL)
- 1 NOT IN (NULL, 1)

# NULL В ЗАПРОСАХ

- 1 NOT IN (0) TRUE
- 1 NOT IN (1) FALSE
- 1 NOT IN (NULL) NULL
- 1 NOT IN (2, NULL) NULL
- 1 NOT IN (NULL, 1) FALSE

# NULL В ЗАПРОСАХ

- IS DISTINCT FROM
  - Аналогично операции ' $\neq$ '
  - Относится к NULL как к известному значению
  - Oracle и MS SQL не умеют ☹️
  - Зато PostgreSQL умеет 😊

# NULL В ЗАПРОСАХ: IS DISTINCT FROM

attr_1	attr_2
1	1
1	2
1	NULL
2	1
2	2
2	NULL
NULL	1
NULL	2
NULL	NULL

```
SELECT attr_1  
        , attr_2  
FROM table_1  
WHERE attr_1 IS DISTINCT FROM attr_2;
```

# NULL В ЗАПРОСАХ: IS DISTINCT FROM

attr_1	attr_2	attr_1 IS DISTINCT FROM attr_2
1	1	FALSE
1	2	TRUE
1	NULL	TRUE
2	1	TRUE
2	2	FALSE
2	NULL	TRUE
NULL	1	TRUE
NULL	2	TRUE
NULL	NULL	FALSE

# NULL В ЗАПРОСАХ: IS DISTINCT FROM

attr_1	attr_2	attr_1 IS DISTINCT FROM attr_2
1	1	FALSE
1	2	TRUE
1	NULL	TRUE
2	1	TRUE
2	2	FALSE
2	NULL	TRUE
NULL	1	TRUE
NULL	2	TRUE
NULL	NULL	FALSE

# NULL В ЗАПРОСАХ: COALESCE

- `COALESCE (attr1, attr2, ... , attrn )`
  - Функция возвращает первое отличное от NULL значение
  - Число аргументов ограничивается только вашей фантазией

# NULL В ЗАПРОСАХ: COALESCE

attr_1	attr_2
1	1
1	2
1	NULL
2	1
2	2
2	NULL
NULL	1
NULL	2
NULL	NULL

```
SELECT coalesce(attr_1, attr_2) AS attr  
FROM table  
WHERE attr_1 IS DISTINCT FROM attr_2;
```



# NULL В ЗАПРОСАХ: COALESCE

attr_1	attr_2
1	1
1	2
1	NULL
2	1
2	2
2	NULL
NULL	1
NULL	2
NULL	NULL



attr
1
1
1
2
2
2
1
2
NULL

# NULL В АГРЕГИРУЮЩИХ ФУНКЦИЯХ

- При подсчетах в агрегирующих функциях `NULL` не учитывается

```
SELECT  sum(attr)
         , min(attr)
         , max(attr)
         , avg(attr)
         , count(attr)
FROM    table;
```

# NULL В АГРЕГИРУЮЩИХ ФУНКЦИЯХ

attr
1
NULL
3
10
2
15
NULL
28
NULL
11
80

```
SELECT  sum(attr)
          , min(attr)
          , max(attr)
          , avg(attr)
          , count(attr)
FROM    table;
```

# NULL В АГРЕГИРУЮЩИХ ФУНКЦИЯХ

attr
1
NULL
3
10
2
15
NULL
28
NULL
11
80



sum(attr)	min(attr)	max(attr)	avg(attr)	count(attr)
150	1	80	18.75	8

# F571: BOOLEAN TEST

- Расширяет оператор `IS` на все значения логического типа данных: `TRUE`, `FALSE`, `NULL`;
- SQL: 1999
- Поддерживается не всеми СУБД, есть в:
  - PostgreSQL
  - MySQL



# РЕЗЮМЕ

- При написании запросов нужно помнить:
  - В какой СУБД вы работаете и на каком диалекте SQL пишете
  - В каких полях вашей таблицы допустимо присутствие NULL
  - Как именно работает троичная логика в разных операциях
  - Корректно ли отработает ваш запрос, если где-то встретится NULL
- Как написать хороший запрос:
  - Сначала подумать
  - Потом написать

# DML: SELECT



```
SELECT [DISTINCT] select_item_comma_list
FROM table_reference_comma_list
[WHERE conditional_expression]
[GROUP BY column_name_comma_list]
[HAVING conditional_expression]
[ORDER BY order_item_comma_list];
```

НЕСКОЛЬКО  
ИСТОЧНИКОВ

---

соединить

---

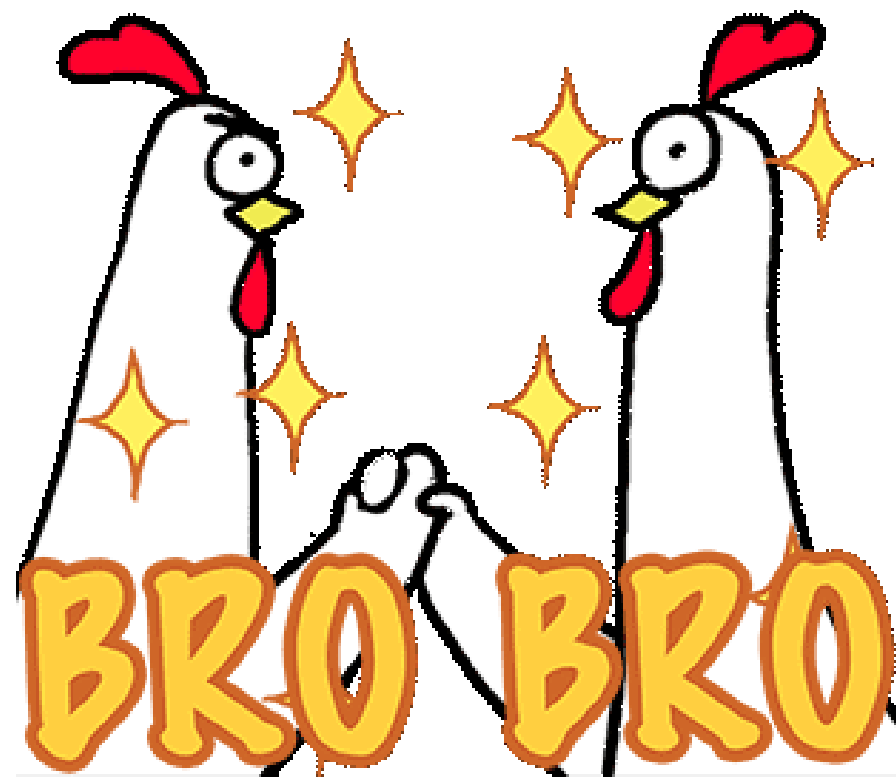
объединить

---

пересечь

---

вычесть





# СОЕДИНЕНИЕ ТАБЛИЦ

- **CROSS JOIN** (полное декартово произведение таблиц)
- **INNER JOIN** (исключает несовпадающие строки)
- **OUTER JOIN** (содержит несовпадающие строки) :
  - **LEFT [OUTER] JOIN**
  - **RIGHT [OUTER] JOIN**
  - **FULL [OUTER] JOIN**

# CROSS JOIN

```
SELECT *  
  FROM table_1  
 CROSS JOIN table_2;
```

```
SELECT column_list_1  
      , column_list_2  
  FROM table_1  
      , table_2;
```

На выходе: полное декартово произведение 2 таблиц

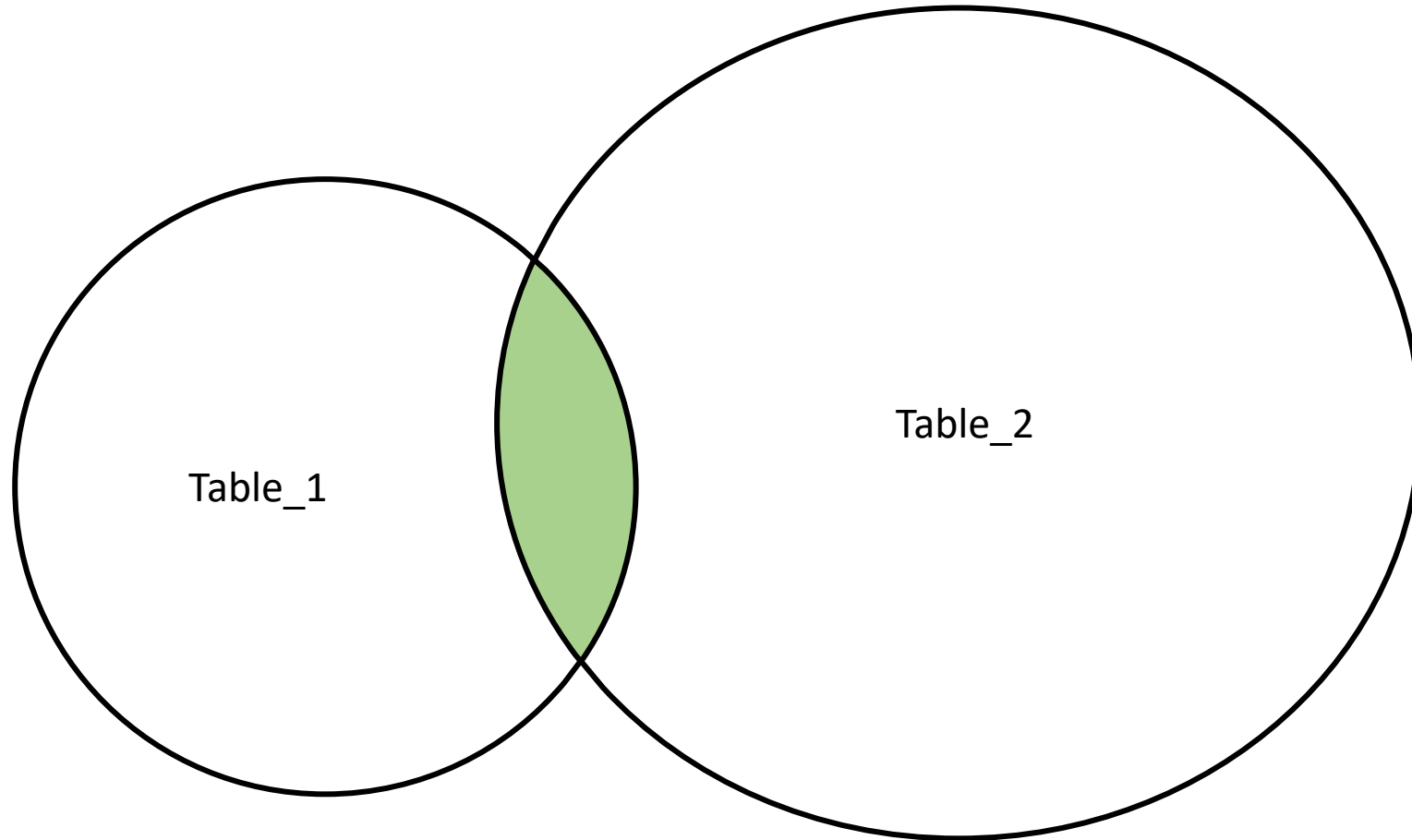
# INNER JOIN

«Сцепление» строк 2 таблиц по заданном условию

```
SELECT column_name_comma_list  
  FROM table_1  
 INNER JOIN table_2  
   ON table_1.column_name = table_2.column_name;
```

Только те строки, для которых условие соединения истинно

# INNER JOIN



# INNER JOIN

PROD_ID	PROD_NAME
123	Computer
124	Laptop
125	Scanner
126	Printer
127	Camera
128	Mobile phone

MANUFACTURER_NAME	PROD_ID
Lenovo	123
Lenovo	124
HP	123
HP	124
HP	125
HP	126
Samsung	128
Lenovo	125
Samsung	123
Samsung	129
LG	129

# INNER JOIN

- Имеем 2 таблицы:
    - PRODUCT (ID продукта, Наименование продукта)
    - PROD\_MAN ( Наименование производителя, ID продукта)
  - Хотим получить связку «продукт-производитель»:
    1. Соединяем таблицы PRODUCT и PROD\_MAN
    2. Оставляем интересующие нас колонки
-

# INNER JOIN

```
SELECT prod_name  
        , manufacturer_name  
FROM product p  
INNER JOIN prod_man pm  
        ON p.prod_id = pm.prod_id;
```

# INNER JOIN

PROD_ID	PROD_NAME
123	Computer
124	Laptop
125	Scanner
126	Printer
127	Camera
128	Mobile phone

MANUFACTURER_NAME	PROD_ID
Lenovo	123
Lenovo	124
HP	123
HP	124
HP	125
HP	126
Samsung	128
Lenovo	125
Samsung	123
Samsung	129
LG	129



# INNER JOIN: РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

PROD_NAME	MANUFACTURER_NAME
Computer	Lenovo
Computer	HP
Computer	Samsung
Laptop	Lenovo
Laptop	HP
Scanner	HP
Scanner	Lenovo
Printer	HP
Mobile phone	Samsung

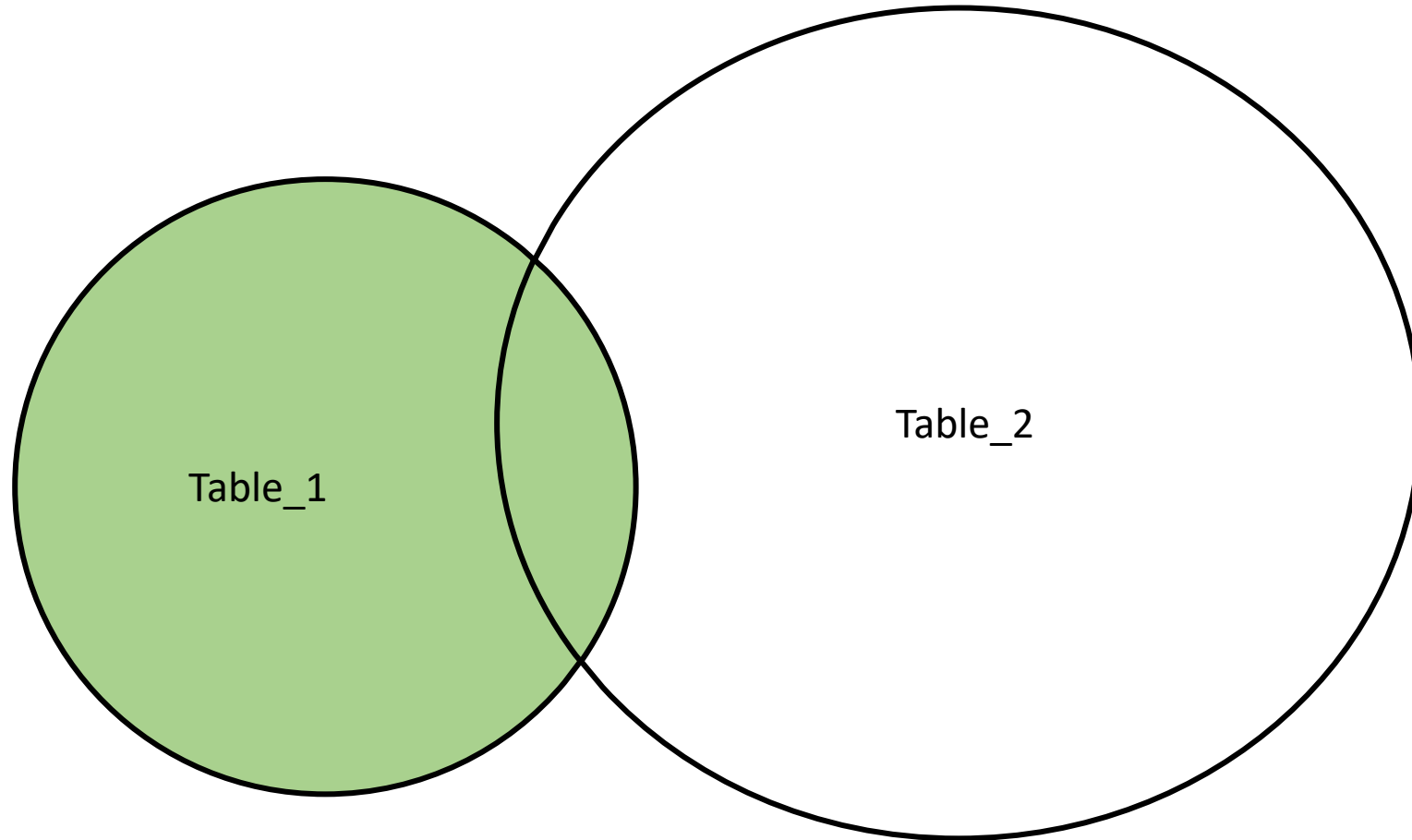
# LEFT JOIN

«Сцепление» строк 2 таблиц по заданном условию

```
SELECT column_name_comma_list
FROM table_1
LEFT JOIN table_2
ON table_1.column_name = table_2.column_name;
```

- В результате присутствуют **все** строки «левой» таблицы
- Те строки, которые не соединяются с «правой» таблицей, все равно попадают в результат
- Поля от «правой» таблицы в таких строках заполняются специальным значением NULL

# LEFT JOIN





# LEFT JOIN

- Задача:
    - Выяснить производителей *всех* продуктов, имеющихся в магазине
  - Решение:
    - Использовать LEFT JOIN вместо INNER JOIN в предыдущей задаче
-

# LEFT JOIN

```
SELECT PROD_NAME  
        , MANUFACTURER_NAME  
FROM PRODUCT P  
LEFT JOIN PROD_MAN PM  
        ON P.PROD_ID = PM.PROD_ID;
```

# LEFT JOIN

PROD_ID	PROD_NAME
123	Computer
124	Laptop
125	Scanner
126	Printer
127	Camera
128	Mobile phone

MANUFACTURER_NAME	PROD_ID
Lenovo	123
Lenovo	124
HP	123
HP	124
HP	125
HP	126
Samsung	128
Lenovo	125
Samsung	123
Samsung	129
LG	129

# LEFT JOIN: РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

PROD_NAME	MANUFACTURER_NAME
Computer	Lenovo
Computer	HP
Computer	Samsung
Laptop	Lenovo
Laptop	HP
Scanner	HP
Scanner	Lenovo
Printer	HP
Camera	NULL
Mobile phone	Samsung

# RIGHT JOIN

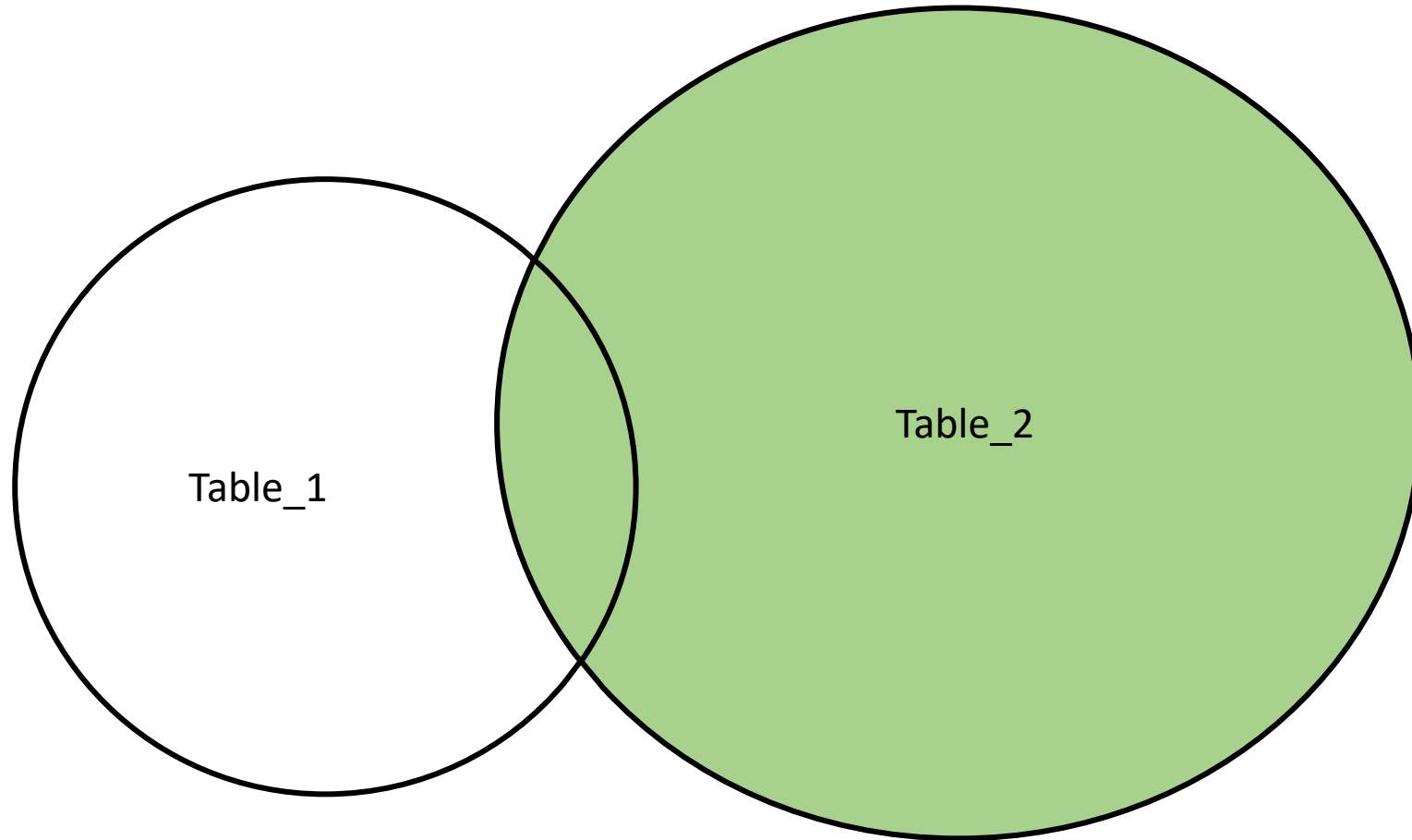
«Сцепление» строк 2 таблиц по заданном условии

```
SELECT column_name_comma_list
FROM table_1
RIGHT JOIN table_2
ON Table1.column_name = Table2.column_name;
```

- В результате присутствуют **все** строки «правой» таблицы
- Те строки, которые не соединяются с «левой» таблицей, все равно попадают в результат
- Поля от «левой» таблицы в таких строках заполняются специальным значением NULL



# RIGHT JOIN





# RIGHT JOIN

- Задача:
    - Выяснить продукты всех производителей, имеющих на складе
  - Решение:
    - Использовать RIGHT JOIN вместо LEFT JOIN в предыдущей задаче
-

# RIGHT JOIN

```
SELECT PROD_NAME  
        , MANUFACTURER_NAME  
FROM PRODUCT P  
RIGHT JOIN PROD_MAN PM  
        ON P.PROD_ID = PM.PROD_ID;
```

# RIGHT JOIN

PROD_ID	PROD_NAME
123	Computer
124	Laptop
125	Scanner
126	Printer
127	Camera
128	Mobile phone

MANUFACTURER_NAME	PROD_ID
Lenovo	123
Lenovo	124
HP	123
HP	124
HP	125
HP	126
Samsung	128
Lenovo	125
Samsung	123
Samsung	129
LG	129

# RIGHT JOIN: РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

PROD_NAME	MANUFACTURER_NAME
Computer	Lenovo
Computer	HP
Computer	Samsung
Laptop	Lenovo
Laptop	HP
Scanner	HP
Scanner	Lenovo
Printer	HP
Mobile phone	Samsung
NULL	Samsung
NULL	LG

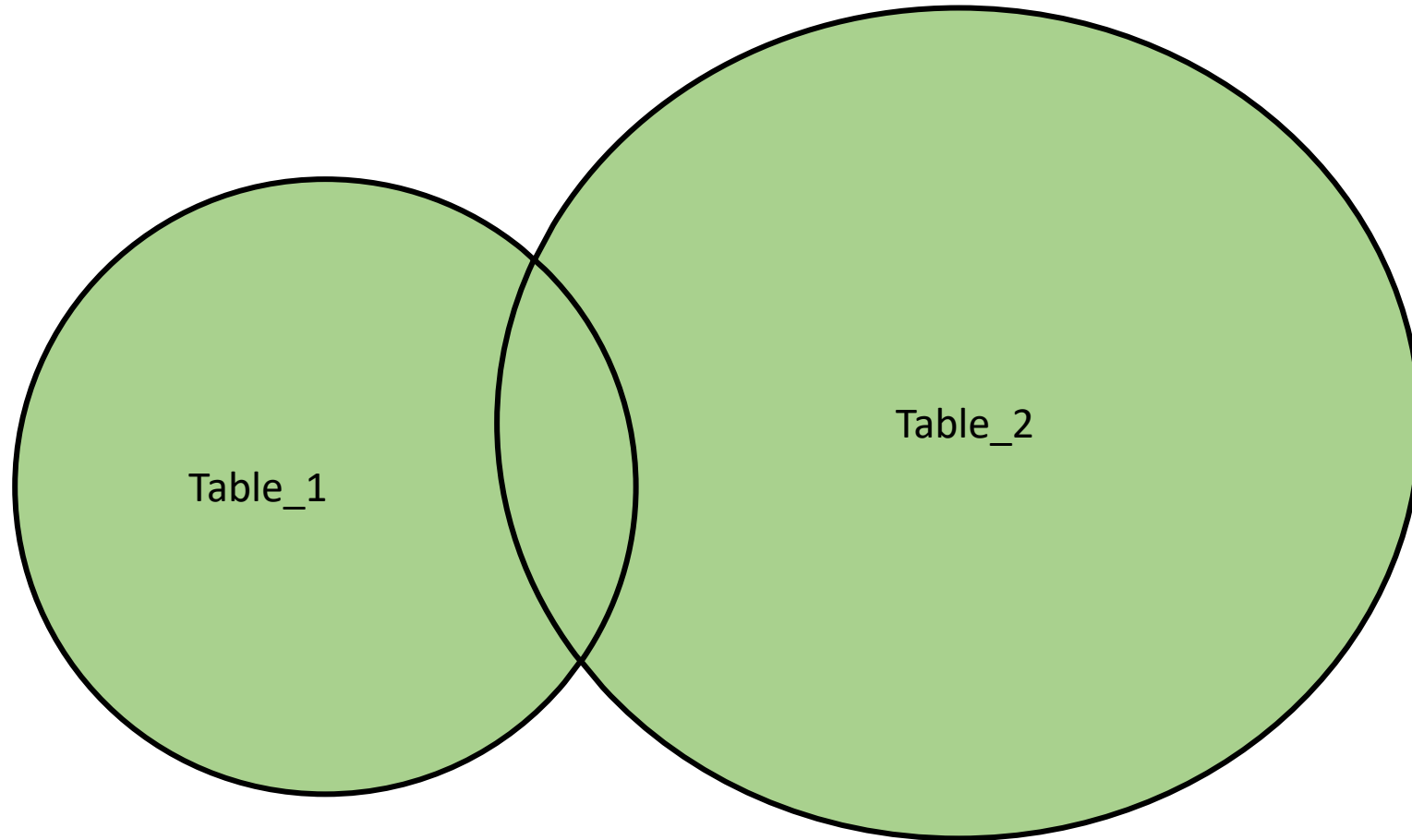
# FULL JOIN

«Сцепление» строк 2 таблиц по заданном условию

```
SELECT column_name_comma_list  
  FROM table_1  
  FULL JOIN table_2  
    ON table_1.column_name = table_2.column_name;
```

- В результирующей таблице присутствуют **все** строки «левой» и «правой» таблиц
- Иными словами, является комбинацией левого и правого соединения

# FULL JOIN



# FULL JOIN

```
SELECT PROD_NAME  
        , MANUFACTURER_NAME  
FROM PRODUCT P  
FULL JOIN PROD_MAN PM  
        ON P.PROD_ID = PM.PROD_ID;
```




# FULL JOIN

PROD_ID	PROD_NAME
123	Computer
124	Laptop
125	Scanner
126	Printer
127	Camera
128	Mobile phone

MANUFACTURER_NAME	PROD_ID
Lenovo	123
Lenovo	124
HP	123
HP	124
HP	125
HP	126
Samsung	128
Lenovo	125
Samsung	123
Samsung	129
LG	129

# FULL JOIN: результат выполнения

PROD_NAME	MANUFACTURER_NAME
Computer	Lenovo
Computer	HP
Computer	Samsung
Laptop	Lenovo
Laptop	HP
Scanner	HP
Scanner	Lenovo
Printer	HP
Camera	NULL
Mobile phone	Samsung
NULL	Samsung
NULL	LG



# SELF-JOIN

- Не является отдельным видом соединения
  - Бывает полезен для некоторого типа задач
  - Задача:
    - Дана таблица RELATIONS (PERSON\_ID, PERSON\_NM, FATHER\_ID)
    - Необходимо сформировать таблицу с именами отцов и детей
-

# SELF-JOIN

PERSON_ID	PERSON_NM	FATHER_ID
1	Иванов Иван Иванович	10
2	Иванов Николай Иванович	1
3	Петров Дмитрий Сергеевич	12
4	Петров Анатолий Дмитриевич	3
5	Петров Петр Дмитриевич	3
6	Степанов Сергей Иванович	13
7	Степанов Матвей Степанович	6
8	Яковлев Андрей Степанович	6

# SELF-JOIN

```
SELECT t2.person_nm AS father_nm
        , t1.person_nm AS son_nm
FROM relations t1 -- дети
INNER JOIN relations t2 -- отцы
ON t1.father_id = t2.person_id;
```

# SELF-JOIN

PERSON_ID	PERSON_NM	FATHER_ID
1	Иванов Иван Иванович	10
2	Иванов Николай Иванович	1
3	Петров Дмитрий Сергеевич	12
4	Петров Анатолий Дмитриевич	3
5	Петров Петр Дмитриевич	3
6	Степанов Сергей Иванович	13
7	Степанов Матвей Степанович	6
8	Яковлев Андрей Степанович	6

# SELF-JOIN: РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ

FATHER_NM	SON_NM
Иванов Иван Иванович	Иванов Николай Иванович
Петров Дмитрий Сергеевич	Петров Анатолий Дмитриевич
Петров Дмитрий Сергеевич	Петров Петр Дмитриевич
Степанов Сергей Иванович	Степанов Матвей Степанович
Степанов Сергей Иванович	Яковлев Андрей Степанович

НЕСКОЛЬКО  
ИСТОЧНИКОВ

---

соединить

---

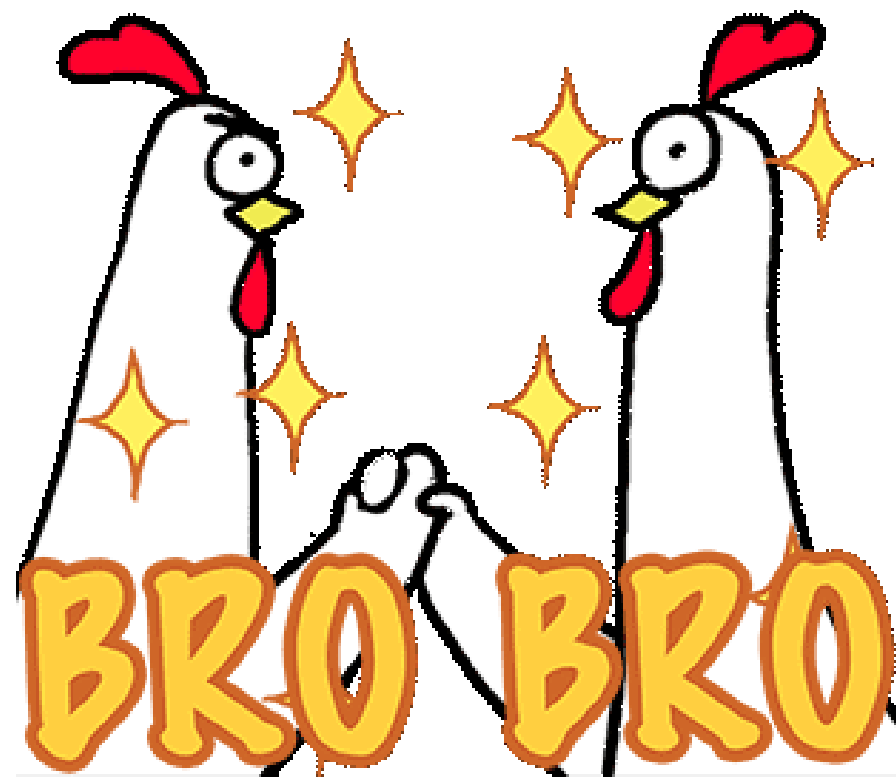
объединить

---

пересечь

---

вычесть





# КОМБИНИРОВАНИЕ НЕСКОЛЬКИХ ИСТОЧНИКОВ

- UNION
- INTERSECT
- EXCEPT

Query\_A

{UNION | INTERSECT | EXCEPT}

Query\_B

# UNION (ALL)

- Позволяет объединять результаты двух и более запросов:
  - С одинаковым числом столбцов
  - Порядок столбцов (семантически) в каждом запросе одинаков
  - Типы данных соответствующих столбцов совместимы
- Результатом является объединение результатов запросов
- Если в Query\_A есть строка X и в Query\_B есть идентичная строка X, то в результате объединения останется только 1 строка X
- Если нужно, чтобы при объединении строка X была записана дважды (или более раз), можно использовать UNION ALL

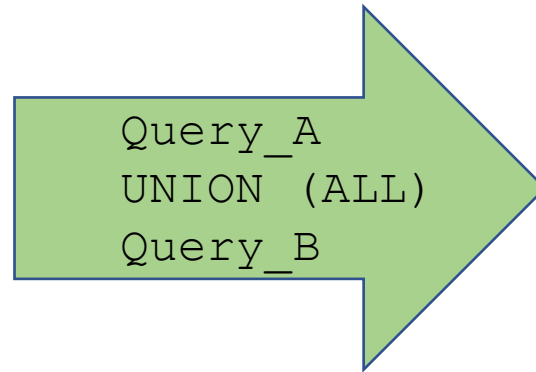
# UNION (ALL)

*Результат выполнения Query\_A*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

*Результат выполнения Query\_B*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина



*Объединение результатов запросов*



# UNION

Результат выполнения Query\_A

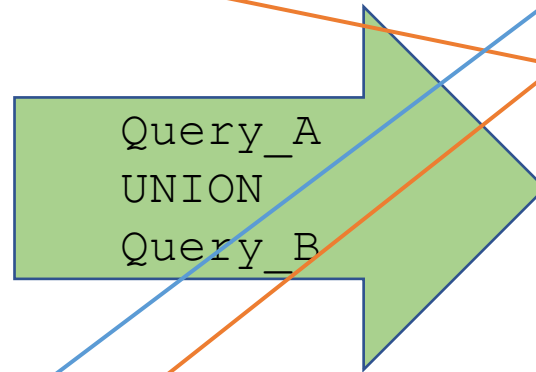
Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

Результат выполнения Query\_B

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина

Объединение результатов запросов

Фамилия	Имя
<b>Меркурьева</b>	<b>Надежда</b>
<b>Халяпов</b>	<b>Александр</b>
Тюрюмина	Элла
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим
Роздухова	Нина



# UNION ALL

Результат выполнения Query\_A

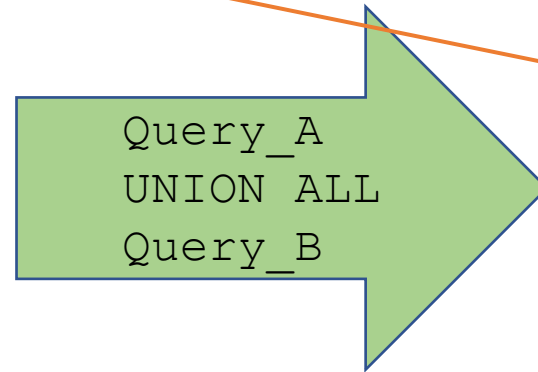
Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

Результат выполнения Query\_B

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина

Объединение результатов запросов

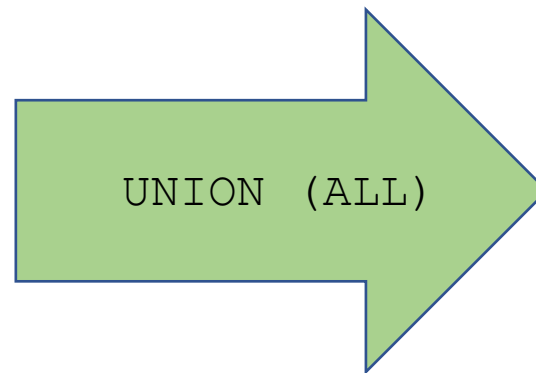
Фамилия	Имя
<b>Меркурьева</b>	<b>Надежда</b>
<b>Халяпов</b>	<b>Александр</b>
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим
<b>Меркурьева</b>	<b>Надежда</b>
<b>Халяпов</b>	<b>Александр</b>
Тюрюмина	Элла
Роздухова	Нина



# UNION (ALL)

Фамилия
Иванов
Петров
Иванов
Сидоров

Фамилия
Иванов
Петров

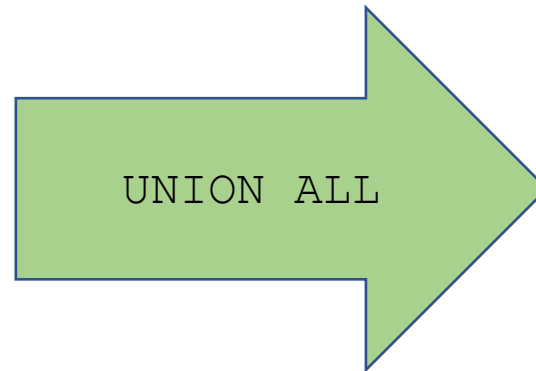


*Объединение результатов запросов*

# UNION ALL

Фамилия
<b>Иванов</b>
Петров
<b>Иванов</b>
Сидоров

Фамилия
<b>Иванов</b>
Петров

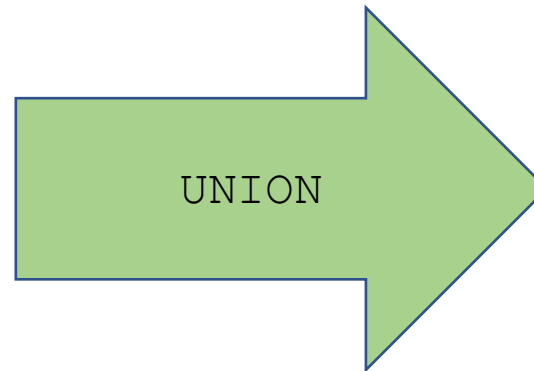


Фамилия
<b>Иванов</b>
Петров
<b>Иванов</b>
Сидоров
<b>Иванов</b>
Петров

# UNION

Фамилия
Иванов
Петров
Иванов
Сидоров

Фамилия
Иванов
Петров



Фамилия
Иванов
Петров
Сидоров



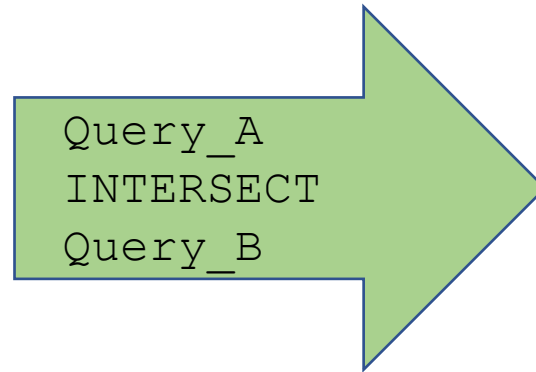
# INTERSECT

*Результат выполнения Query\_A*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

*Результат выполнения Query\_B*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина



*Объединение результатов запросов*

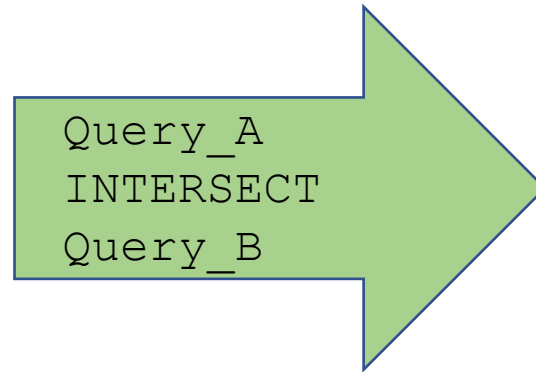
# INTERSECT

*Результат выполнения Query\_A*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

*Результат выполнения Query\_B*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина



*Пересечение результатов запросов*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр

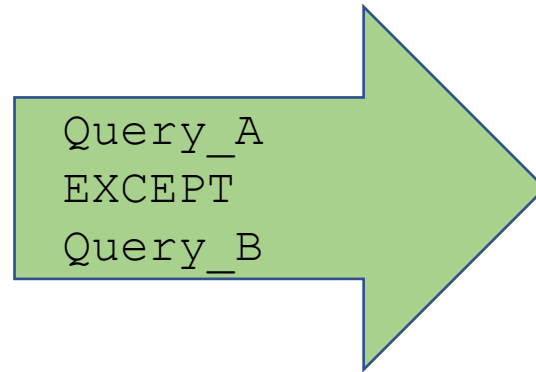
# EXCEPT

*Результат выполнения Query\_A*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

*Результат выполнения Query\_B*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина



*Разность результатов запросов*



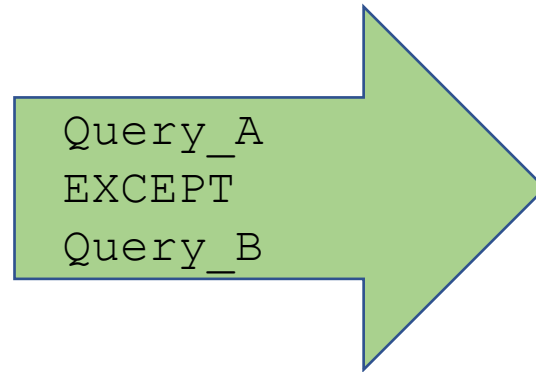
# EXCEPT

*Результат выполнения Query\_A*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

*Результат выполнения Query\_B*

Фамилия	Имя
Меркурьева	Надежда
Халяпов	Александр
Тюрюмина	Элла
Роздухова	Нина



*Разность результатов запросов*

Фамилия	Имя
Мавлютов	Максим
Лукьянов	Денис
Митюрин	Максим

# NULL В ЗАПРОСАХ

- Для теоретико-множественных операций
  - UNION
  - INTERSECT
  - EXCEPT

NULL = NULL

- **НО!** Для операции UNION ALL NULL <> NULL

# SQL ЗАПРОС: БУДЕТ ЛИ РАБОТАТЬ?

```
SELECT t1.attr1_1, ..., t1.attr1_N, t2.attr2_1, ..., t2.attr2_M
FROM t1
INNER|LEFT|RIGHT|FULL JOIN t2
    ON t1.attr1_1 = t2.attr2_1 AND t1.attr1_N > t2.attr2_M
WHERE t1.attr1_k BETWEEN X AND Y
    OR t2.attr2_1 LIKE 'a%'
GROUP BY t1.attr1_2, t2.attr2_2
HAVING SUM(t1.attr1_3) > 0
UNION
SELECT t3.attr3_1, ..., t3.attr3_N, t4.attr4_1, ..., t4.attr4_M
FROM t3
INNER|LEFT|RIGHT|FULL JOIN t4
    ON t3.attr3_1 > t4.attr4_2;
```

# СВЯЗЬ РЕЛЯЦИОННОЙ АЛГЕБРЫ С SQL

Реляционная алгебра	SQL
<b>Теоретико-множественные операции</b>	
Объединение	UNION
Пересечение	INTERSECT
Разность	EXCEPT
<b>Реляционные операции</b>	
Ограничение	WHERE
Проекция	SELECT
Соединение	JOIN
Деление	-

# ПРИМЕРЫ ЗАПРОСОВ

Реляционная алгебра	SQL
<b>Из таблицы Product(product_id, product_name, product_type, product_price) выделить все наименования продуктов с ценой не менее 150 рублей</b>	
P150 = Product[product_price >= 150] R = P150[product_name] Ответ: R.	<b>SELECT DISTINCT</b> product_name <b>FROM</b> product <b>WHERE</b> product_price >= 150;
<b>Пусть даны таблицы Product(product_id, product_name, product_type, product_price) и Shop(shop_id, shop_name, product_id). Найти все магазины, в которых продается мыло с ценой менее 150 рублей</b>	
PS = Product[product_type = 'Мыло'][product_price < 150] SS = PS[PS[product_id] = Shop[product_id]]Shop SSN = SS[shop_name] Ответ: SSN.	<b>SELECT DISTINCT</b> shop_name <b>FROM</b> product p <b>INNER JOIN</b> shop sh <b>ON</b> p.product_id = sh.product_id <b>WHERE</b> product_type = 'Мыло' <b>AND</b> product_price < 150;





Вопросы?