/*

Курс по СУБД **Oracle:**

основы администрирования, **SQL**, **PL/SQL**

CREATE OR REPLACE PACKAGE "Занятие 3" AS

*/

T · · Systems ·

План занятия

- Групповые функции
- **`GROUP BY`** и **`HAVING`**
- Типы связей/отношений
- Соединения таблиц (**`JOIN`**) и их разновидности
- Подзапросы
- Работа с множествами

`END;`

T · · Systems ·

```
CREATE OR REPLACE PACKAGE BODY “Занятие 3”
AS
   l_alert VARCHAR2(10);
BEGIN
   l_alert := ‘Продолжаем?’;
   dbms_output.put_line(l_alert);
```

Групповые функции

Обрабатывают **много** строк — возвращают **один**

```
SELECT group_function(column),…
  FROM table;
```

Игнорируют **NULL** — его нужно обрабатывать, например, **NVL**:

- **COUNT**({*|выражение}) — число строк в группе
- **AVG** — среднее значение
- **SUM** — сумма значений
- **MAX**, **MIN** — максимальное и минимальное значение в группе

GROUP BY

```
SELECT column, group_function(column)
  FROM table
[WHERE condition]
[GROUP BY group_by_expression];
```

Создание групп данных, внутри каждой из которых будет применяться групповая функция.

Особенности:

- Необходимо указывать условие группировки для всех столбцов в **SELECT**, кроме тех, к которым применяется функция
- Для групповых функций нельзя использовать условие **WHERE**

T · · Systems ·

HAVING

```
 SELECT column, group_function(column)
   FROM table
 [WHERE condition]
 [GROUP BY group_by_expression]
[HAVING group_condition];
```

Позволяет исключать какие-то группы данных.

Порядок работы:

1. Строки группируются по какому-то выражению
2. Применяется групповая функция внутри каждой группы
3. Выводятся группы, удовлетворяющие условию **HAVING**

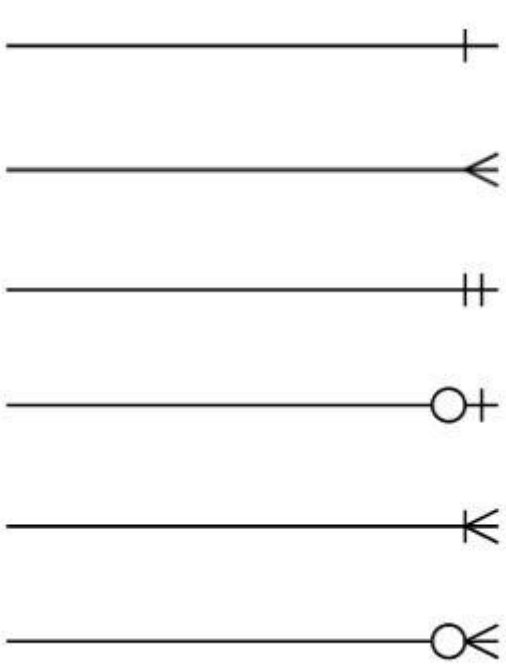
T··Systems·

HAVING
Пример

```
-- Вывести список МВЗ со списком зарезервированных на них
мест, если их больше пяти; указать также город, корпус и этаж
SELECT ct.name                  AS город,
       o.name                   AS корпус,
       f.name                   AS этаж,
       c.reservation            AS мвз,
       COUNT(c.addr)            AS количество,
       LISTAGG(c.addr, '|') AS места
  FROM coords c, floors f, offices o, cities ct
 WHERE c.floor_id = f.id
   AND f.office_id = o.id
   AND o.city_id = ct.id
   AND c.employee_id = 0
   AND reservation IS NOT NULL
 GROUP BY ct.name, o.name, f.name, c.reservation
HAVING COUNT(c.addr) > 5
 ORDER BY ct.ord DESC, o.ord DESC, f.ord ASC, c.reservation;
```

Отношения в реляционных БД

- «Один ко многим»
- «Многие ко многим»
- «Один к одному»



«один»

«много»

«один и только один»

«ноль или один»

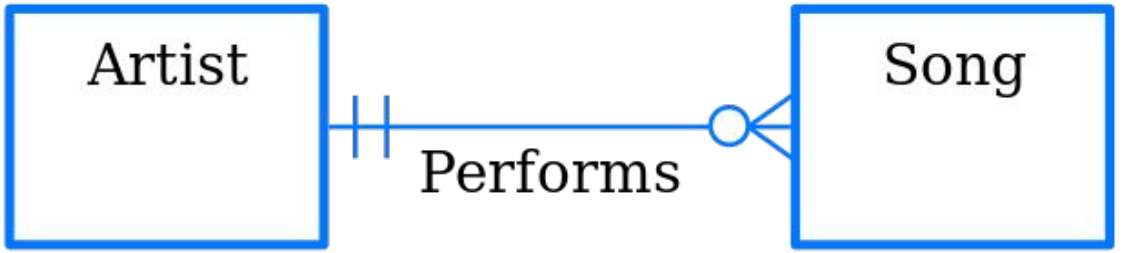
«один или много»

«ноль или много»

T · · Systems ·

«Один ко многим» ”one-to-many”

«У строки таблицы А может быть несколько совпадающих строк таблицы Б, но каждой строке таблицы Б может соответствовать только одна строка из А»



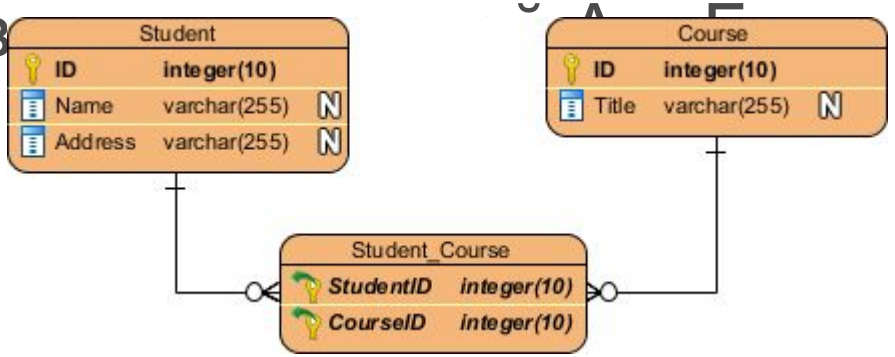
Пример:

Каждый артист может быть быть исполнителем нескольких, одной или многих песен, но у каждой песни может быть один и только один исполнитель.

T · · Systems ·

«Многие ко многим» “many-to-many”

«Строке таблицы А может сопоставляться несколько строк таблицы Б, и наоборот. Для создания этой связи нужна третья таблица — “таблица соединения”, — чей первичный ключ состоит из в



Пример:

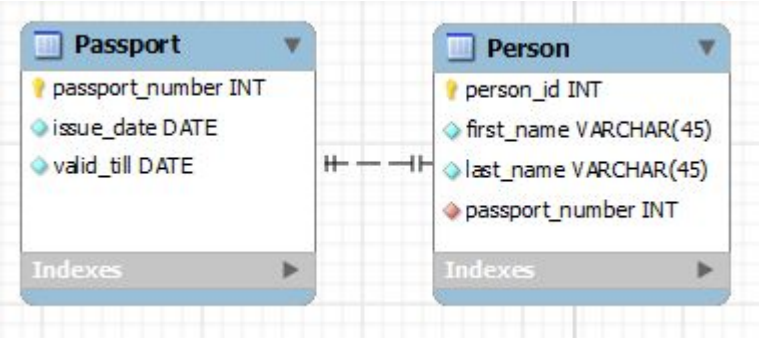
Каждый студент может посещать один или несколько курсов или не посещать ничего. На каждом курсе может быть один и больше студентов, или набора на него не было, поэтому там нет студентов.

T · · Systems ·

«Один к одному» “one-to-one”

«Строке таблицы А может сопоставляться только одна строка таблицы Б, и наоборот»

Используется, напрмиер, для изоляции части таблицы из соображений безопасности или хранения данных, которые можно легко удалить.



Пример:

У человека может быть один и только один паспорт, ровно как и один паспорт может идентифицировать одного и только одного человека.

T · · Systems ·

Типы соединений

- Декартово произведение (**CROSS JOIN**)
- Внутреннее (простое) соединение (**EQUI/INNER JOIN**)
- Соединение по неравенству (**NON EQUI JOIN**)
- Естественное соединение (**NATURAL JOIN**)
- Самообъединение (**SELF JOIN**)
- Внешнее соединение (**OUTER JOIN**)
  - Левое внешнее соединение (**LEFT [OUTER] JOIN**)
  - Правое внешнее соединение (**RIGHT [OUTER] JOIN**)
  - Полное внешнее соединение (**FULL [OUTER] JOIN**)

Синтаксис соединений Oracle

```
SELECT table1.column, table2.column
  FROM table1
       [NATURAL JOIN table2] |

      [[INNER]JOIN table2
        USING (column_name)] |

      [[INNER]JOIN table2
        ON table1.column_name = table2.column_name]|

       [LEFT|RIGHT|FULL [OUTER] JOIN table2
        ON table1.column_name = table2.column_name]|

       [CROSS JOIN] table2];
```

Декартово произведение
CROSS JOIN

Соединение, при котором все строки первой таблицы объединяются со всеми строками второй таблицы.

Получается, если условие соединение **опущено** или **не действует**

```
SELECT *
  FROM customers, addresses;
```

```
SELECT *
  FROM customers
       CROSS JOIN addresses;
```

T · · Systems ·

Декартово произведение
CROSS JOIN

```
-- 599
SELECT COUNT(*)
  FROM customers;

-- 603
SELECT COUNT(*)
  FROM addresses;

-- 599 x 603 = 361197
SELECT COUNT(*)
  FROM customers
       CROSS JOIN addresses;
```

T · · Systems ·

Внутреннее соединение
INNER JOIN

Соединение, при котором только те строки первой таблицы объединяются со строками второй таблицы, у которых совпадает с ними значение.

```
SELECT *
  FROM customers, addresses
 WHERE cust_address_id = addr_id;
```

ИЛИ

```
SELECT *
  FROM customers
       [INNER] JOIN addresses
                  ON cust_address_id = addr_id;
```

T · · Systems ·

Соединение по неравенству “NON EQUI JOIN”

Соединение, противоположное к внутреннему; объединяются только такие строки двух таблиц, которые удовлетворяют указанному условию неравенства: <>, >, <, **BETWEEN** и т.д.

COUNT(CROSS JOIN) = COUNT(INNER JOIN) + COUNT(NON EQUI JOIN)

Естественное соединение
NATURAL JOIN

- Соединение, основанное на всех столбцах, имеющих одинаковые названия
- Строки выбираются в этих столбцах тогда, когда их значения и типы данных в одинаковых столбцах совпадают
- В отличие от **INNER JOIN**'а **NATURAL JOIN** в результирующей выборке не будет возвращать

```
SELECT *
  FROM customers
       NATURAL JOIN addresses;
```

T · · Systems ·

USING

- У столбцов соединяемых таблиц одинаковые имена, но разные типы данных
- В таблицах есть несколько столбцов с одинаковыми именами и типами, но соединение должно быть не по всем

```
SELECT *
  FROM customers
       JOIN addresses
      USING (addr_id);
```

T · · Systems ·

ON

- Необходимо задать конкретные столбцы, по которым осуществлять соединение
- Имена столбцов в соединяемых таблицах не обязательно должны быть одинаковыми

```
SELECT *
  FROM customers c
       JOIN addresses a
         ON c.cust_addr_id = a.addr_id;
```

T · · Systems ·

Самообъединение
“SELF JOIN”

- Соединение таблицы самой с собой
- Для того, чтобы недвусмысленно определить столбцы для условия, обязательно использование псевдонимов для таблиц

```
SELECT *
  FROM employees e, employees m
 WHERE e.manager_id = m.employee_id;
```

T · · Systems ·

Внешнее соединение
OUTER JOIN

- Соединяет таблицы по указанным столбцам, выводя строки, удовлетворяющие и не удовлетворяющие условию
- В Oracle можно использовать упрощённый оператор (+) для выполнения внешнего соединения, поставив его там, где возможно отсутствие значений для

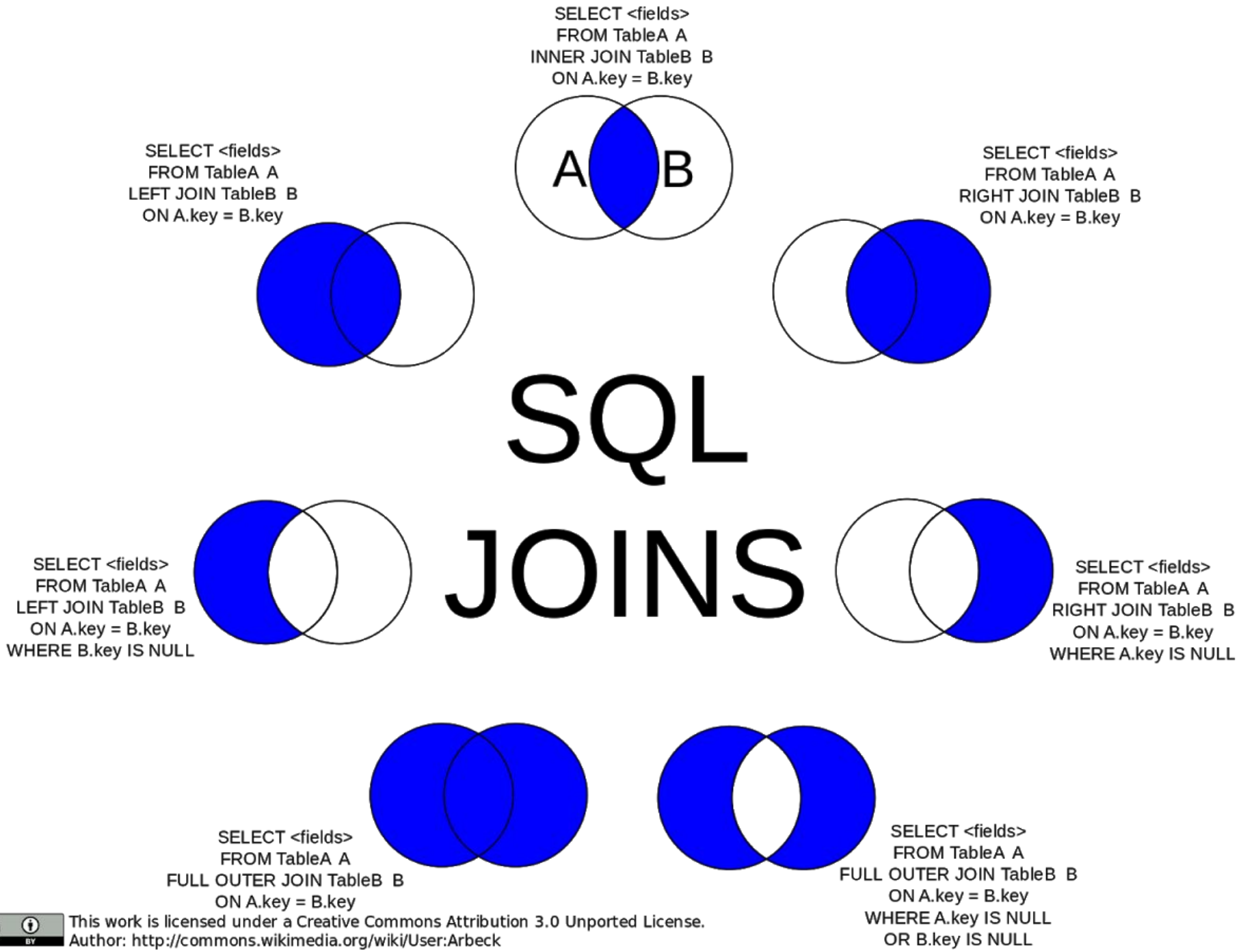
```
-- Будут выведены все сотрудники с соответствующими отделами,
а также и те сотрудники, которые не числятся в отделах
SELECT *
  FROM employees e, department d
 WHERE e.dept_id = d.dept_id(+);
```

T · · Systems ·

Внешнее соединение
OUTER JOIN

- ANSI-синтаксис подразумевает указание типа внешнего соединения
- **FULL OUTER JOIN** возможен только в ANSI-синтаксисе
- Oracle **не** рекомендует использовать специфичный

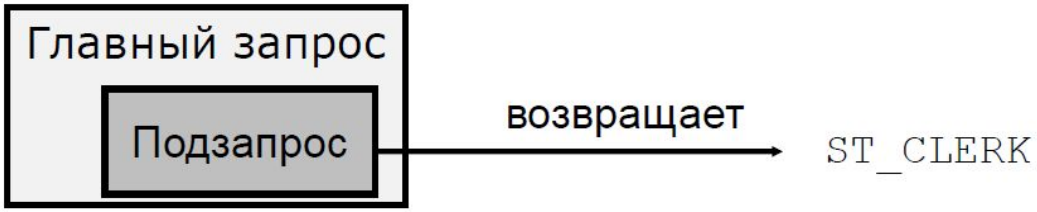
```
SELECT *
  FROM employees e
       LEFT OUTER JOIN department d
                    ON e.dept_id = d.dept_id;
```



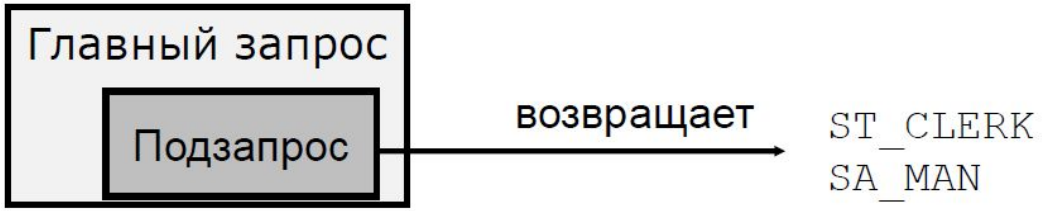
T · · Systems ·

Подзапросы

- Однострочные (скалярные)



- Многострочные (табличные)




```
SELECT select_list
  FROM table
 WHERE expr_operator
                    (SELECT select_list
                       FROM table);
```

T · · Systems ·

Скалярные подзапросы

- Возвращают только одну строку
- Используются с однострочными операторами сравнения

```
-- Вывести список сотрудников той же должности, что и у Пупкина, у
которых зарплата больше, чем у него
SELECT last_name, salary
  FROM employees
 WHERE job_id =
                (SELECT job_id
                   FROM employees
                  WHERE last_name = 'Pupkin')
   AND salary >
                (SELECT salary
                   FROM employees
                  WHERE last_name = 'Pupkin');
```

Табличные подзапросы

- Возвращают больше одной строки
- Используются с операторами множественного сравнения
  - **IN/NOT IN** — равно одному значению/не равно ни одному значению из списка
  - **ANY/SOME** — сравнивает значения с каждым в списке

```
SELECT product_name
  FROM products
 WHERE product_id = ANY (SELECT product_id
                           FROM order_details
                          WHERE quantity > 10);
```

Ничего не замечаете?

= ANY тождественно **IN**

Ещё про подзапросы

- Можно использовать строковые конструкции

```
SELECT * FROM t1
WHERE ROW(col1, col2) = (SELECT col3, col4 FROM t2 WHERE id = 10);
```

Оператор **ROW** указывать не обязательно

- Можно формировать подзапросы с помощью ключевых слов **EXISTS** и **NOT EXISTS**

```
SELECT DISTINCT s.store_type
  FROM stores s
 WHERE EXISTS (SELECT *
                 FROM cities_stores cs
                WHERE cs.store_type = s.store_type);
```

- Подзапросы можно использовать в операторе **FROM**
- Чаще всего **JOIN** более оптимизированно решают задачу

T · · Systems ·

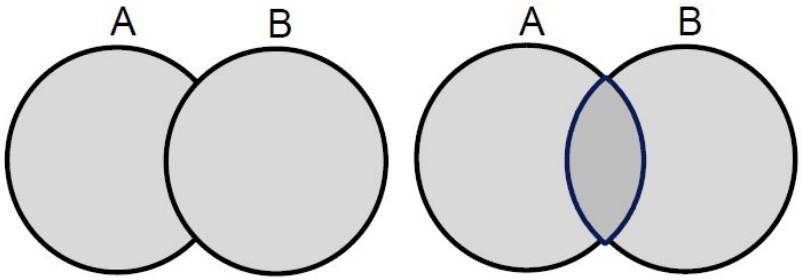
Работа с множествами

Особенности:

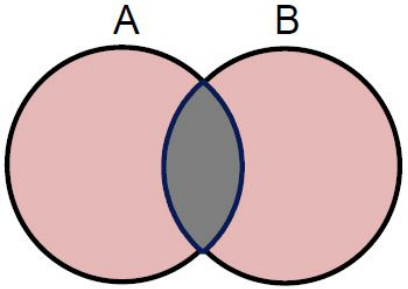
- Число столбцов в запросах должно быть одинаковым
- Типы данных столбцов в запросах должны быть идентичны
- При помощи скобок можно задать порядок объединения запросов

Работа с множествами

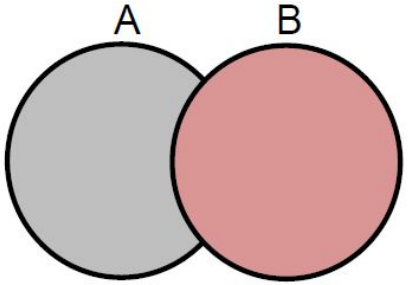
- Объединение: **UNION** **UNION ALL**



- Пересечение: **INTERSECT**



- Вычитание: **EXCEPT/MINUS**



T · · Systems ·

UNION

- Объединение набора результатов из двух и более **SELECT**'ов
- **UNION** удаляет повторящиеся строки

```
-- Вывести список клиентов со всеми датами, когда они совершали
покупки или продажи
SELECT customer_id, purchase_date AS “date”
  FROM purchases

 UNION ALL

SELECT customer_id, sale_date
  FROM sales;
```

T · · Systems ·

INTERSECT

- Возвращает строки, выбранные для всех запросов
- Если запись существует в результатах одного запроса, а в результатах другого нет, то она будет исключена из конечного результата

```
-- Вывести список людей с указанием их ФИО, а также персонажа,
которого они играли
SELECT last_name, first_name, character
  FROM actors

INTERSECT

SELECT last_name, first_name, NULL
  FROM producers;
```

MINUS

- Возвращает все строки первого **SELECT**'а, которые не возвращает второй **SELECT**

```
-- Вывести ID сотрудников, которые не являются администраторами
SELECT employee_id
  FROM employees

 MINUS

SELECT employee_id
  FROM admins;
```

Ещё про работу с множествами

- Если количество столбцов запросов не совпадает, можно искусственно создать псевдостолбцы
- **`ORDER BY`** указывается только в последнем **`SELECT`**’е
- В **`ORDER BY`** указываются столбцы из первого **`SELECT`**’а
- По умолчанию сортировка происходит во всех операторах (кроме **`UNION ALL`**) в порядке возрастания
- Названия столбцов результирующей выборки даются такие же, какие указаны в первом **`SELECT`**’е

T · · Systems ·

Практика

1. Создать в своей схеме представление HW3_1 со списком со следующими полями:
   - «**Клиент**» — нормализованные ФИО (например, «Иван Иванов»)
   - «**Заказов**» — количество заказов, совершённых покупателем за всё время
   - «**Бонус**» — указывается в долларах и рассчитывается исходя из того, что за один заказ начисляется бонус 18¢ (например, за 60 заказов бонус будет «$10.80»)

   Бонусы начисляются только тем, кто совершил больше 30 заказов. Список вывести в порядке убывания выплат клиентам.
2. Создать в своей схеме представление HW3_2 со списком email'ов («**Email**») только тех как сотрудников, так и покупателей, по адресу проживания которых живёт не только один человек.

T · · Systems ·

```
  EXCEPTION
    WHEN questions
      just_ask;
    WHEN others
      NULL;
  END;
END "Занятие 3";

ALTER PACKAGE "Занятие 3" COMPILE BODY;
ALTER PACKAGE "Занятие 3" COMPILE PACKAGE;
```