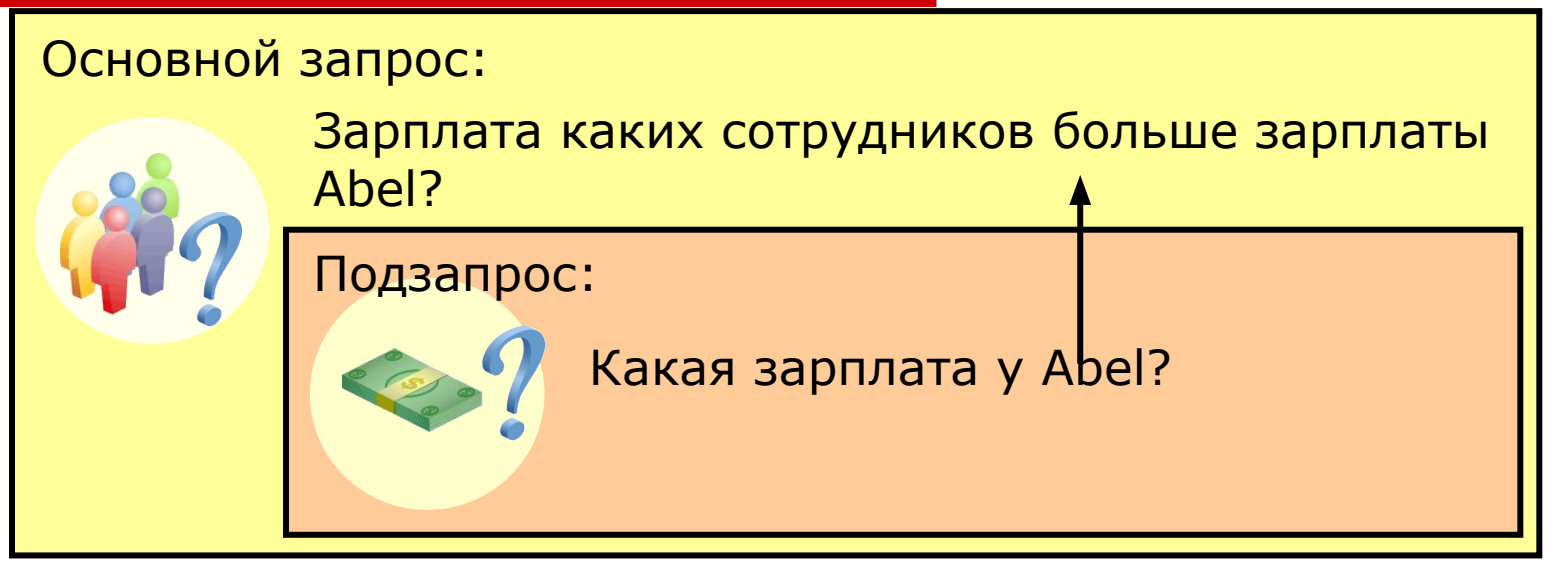
Использование подзапросов

Использование подзапросов для решения задач



- Внутренний запрос (или подзапрос) возвращает значение, которое используется внешним запросом (или основным запросом).
- Использование подзапроса эквивалентно выполнению двух последовательных запросов и применения результата первого запроса в качестве значения для поиска во втором запросе.

Синтаксис подзапроса

Подзапрос - оператор SELECT, встраиваемый в предложение другого оператора SELECT.

Подзапросы можно размещать в :

- В предложении WHERE
- В предложении HAVING
- В предложении FROM

```
SELECT select_list
FROM   table
WHERE  expr operator
           (SELECT select_list
              FROM   table);
```

Параметр operator может быть как однострочным оператором (>, =, >=, <, <>, <=), так и многострочным оператором (IN, ANY, ALL, EXISTS).

- Подзапрос (внутренний запрос) выполняется перед основным запросом (внешним запросом).
- Результат подзапроса используется основным запросом.

Пример использования подзапросов

```
SELECT last_name, salary
FROM   employees
WHERE  salary >    1100
                   0
                  (SELECT salary
                   FROM   employees
                   WHERE  last_name = 'Abel');
```

	LAST_NAME	SALARY
1	King	24000
2	Kochhar	17000
3	De Haan	17000
4	Hartstein	13000
5	Higgins	12000

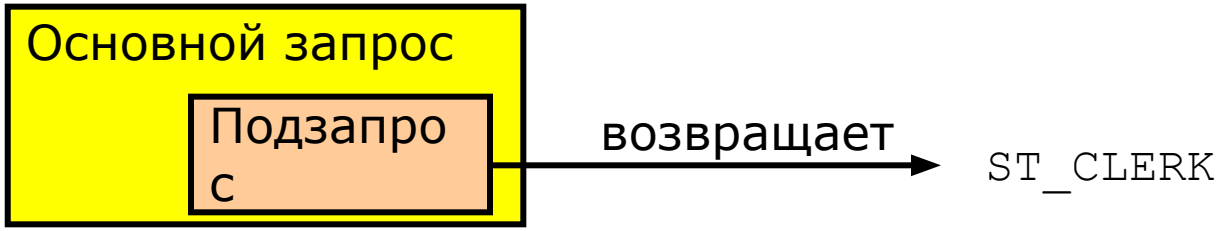
Внешний запрос использует результат внутреннего запроса для вывода на экран всех сотрудников, зарплата которых больше, чем сотрудника Абеля.

Рекомендации по составлению подзапросов

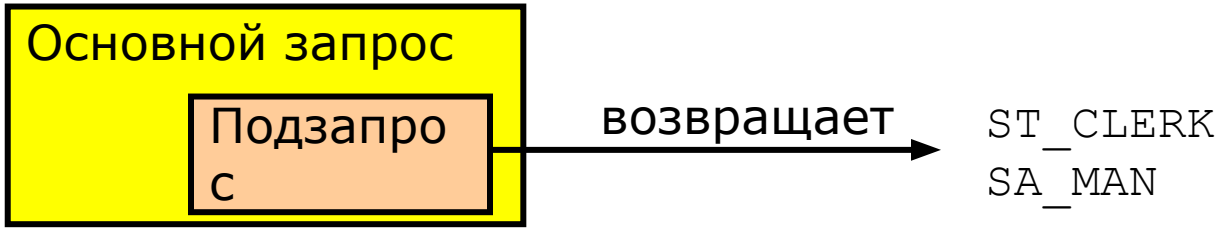
- Подзапросы необходимо заключать в круглые скобки.
- Для лучшей читаемости рекомендуется располагать подзапрос в правой части условия сравнения.
- Если результат подзапроса возвращает одно значение, то используются однострочные операторы для работы с этим результатом.
- Если результатом запроса может быть множество значений, то используются многострочные операторы для обработки результатов таких подзапросов.

Типы подзапросов

- Однострочные подзапросы - запросы, возвращающие только одну строку из внутреннего оператора SELECT.



- Многострочные подзапросы - запросы, возвращающие несколько строк из внутреннего оператора SELECT.



- Многостолбцовые подзапросы - подзапросы, возвращающие больше чем один столбец из внутреннего оператора SELECT.

Однострочные подзапросы

- Возвращают только одну строку.
- Используются с однострочными операторами сравнения:

Оператор	Значение
=	Равно
>	Больше
>=	Больше или равно
<	Меньше
<=	Меньше или равно
<>	Не равно

Выполнение однострочных подзапросов

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Taylor')
AND    salary >
                (SELECT salary
                 FROM   employees
                 WHERE  last_name = 'Taylor');
```

SA_REP

8600

	LAST_NAME	JOB_ID	SALARY
1	Abel	SA_REP	11000

Использование групповых функций в подзапросах

```
SELECT last_name, job_id, salary
FROM   employees
WHERE  salary =
                (SELECT MIN(salary)
                 FROM   employees);
```

250

0

	LAST_NAME	JOB_ID	SALARY
1	Vargas	ST_CLERK	2500

В результате запроса выбираются фамилия, ID работы и заработная плата сотрудников, для которых она равна минимальной заработной плате. Групповая функция MIN возвращает единственное значение (2500), передаваемое внешнему запросу.

Использование подзапросов в условии HAVING

- Сервер Oracle выполняет сначала подзапросы.
- Результат подзапроса передается в инструкцию HAVING основного запроса.

```
SELECT    department_id, MIN(salary)
FROM      employees
GROUP BY  department_id
HAVING    MIN(salary) > (SELECT MIN(salary)
                         FROM   employees
                         WHERE  department_id = 50);
```

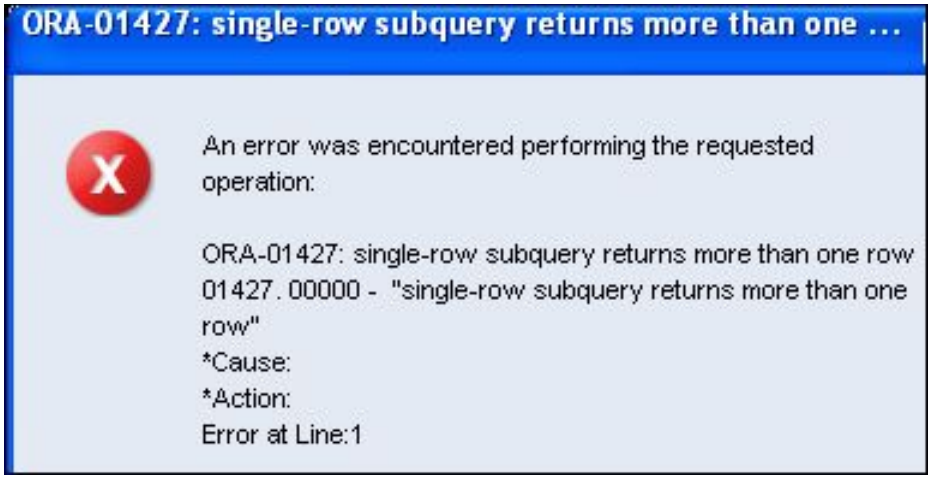
250

0

	DEPARTMENT_ID	MIN(SALARY)
1	(null)	7000
2	90	17000
3	20	6000
...		
7	10	4400

Тест

```
SELECT employee_id, last_name
FROM   employees
WHERE  salary =
                (SELECT   MIN(salary)
                 FROM     employees
                 GROUP BY department id);
```



Подзапрос не возвращает строк

```
SELECT last_name, job_id
FROM   employees
WHERE  job_id =
                (SELECT job_id
                 FROM   employees
                 WHERE  last_name = 'Haas');
```

```
0 rows selected
```

Подзапрос не возвращает строк, так как не существует сотрудника с именем “Haas”, поэтому основной запрос также не возвращает значений.

Многострочные подзапросы

- Возвращают больше чем одну строку.
- Используются с многострочными операторами сравнения .

Оператор	Значение
IN	Равен любому элементу из списка
ANY	Перед оператором должен быть оператор =, !=, >, <, <=, >=. Сравнивает значение с каждым значением из списка, возвращаемого подзапросом. Возвращает FALSE, если подзапрос не возвращает ни одной строки.
ALL	Перед оператором должен быть оператор =, !=, >, <, <=, >=. Сравнивает значение с каждым значением из списка, возвращаемого подзапросом. Возвращает TRUE, если подзапрос не возвращает ни одной строки.

Использование оператора ANY в многострочных подзапросах

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ANY
               (SELECT salary
                FROM   employees
                WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	144	Vargas	ST_CLERK	2500
2	143	Matos	ST_CLERK	2600
3	142	Davies	ST_CLERK	3100
4	141	Rajs	ST_CLERK	3500
5	200	Whalen	AD_ASST	4400
...				
9	206	Gietz	AC_ACCOUNT	8300
10	176	Taylor	SA_REP	8600

- <ANY - меньше, чем максимальное значение из набора.
- >ANY - больше, чем минимальное значение из набора.
- =ANY - эквивалентно IN.

Использование оператора ALL в многострочных подзапросах

```
SELECT employee_id, last_name, job_id, salary
FROM   employees
WHERE  salary < ALL
                (SELECT salary
                 FROM   employees
                 WHERE  job_id = 'IT_PROG')
AND    job_id <> 'IT_PROG';
```

9000, 6000, 4200

- <ALL - меньше, чем минимальное значение из набора.
- >ALL -больше, чем максимальное значение из набора.

	EMPLOYEE_ID	LAST_NAME	JOB_ID	SALARY
1	141	Rajs	ST_CLERK	3500
2	142	Davies	ST_CLERK	3100
3	143	Matos	ST_CLERK	2600
4	144	Vargas	ST_CLERK	2500

NULL-значения в подзапросе и оператор NOT IN

Задача: отобразить всех сотрудников, которые не являются менеджерами.

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id NOT IN
                        (SELECT
mgr.manager_id
                         FROM   employees
mgr);
```

0 rows selected

По логике запрос должен отображать сотрудников (12 сотрудников), однако запрос возвращает 0, т.к. одно из значений внутреннего подзапроса = NULL.

Оператор NOT IN равносилен оператору <>ALL, т.е. сравниваются значения на больше/меньше, однако если множество значений, возвращаемое подзапросом, содержит неопределенное значение NULL, то запрос возвращает 0 строк (пустое множество), поскольку невозможно сравнить конкретную величину с неопределенным значением.

NULL-значения в подзапросе и оператор IN

Задача: отобразить всех сотрудников, которые являются менеджерами.

```
SELECT emp.last_name
FROM   employees emp
WHERE  emp.employee_id  IN
                        (SELECT mgr.manager_id
                         FROM   employees mgr);
```

Данный запрос выполнится успешно, т.к. IN эквивалентен =ANY, а проверка на равенство значению NULL является допустимой.

NULL-значения в подзапросе оператор NOT IN и NOT NULL

Задача: отобразить всех сотрудников, которые не являются менеджерами.

Задачу можно решить, если в подзапрос добавить проверку на наличие NULL–значений в результате и удалить данные строки из результата подзапроса.

```
SELECT last_name
FROM employees
WHERE employee_id NOT IN
          (SELECT manager_id
           FROM employees
           WHERE manager_id IS NOT NULL);
```

Операторы множеств - SET операторы

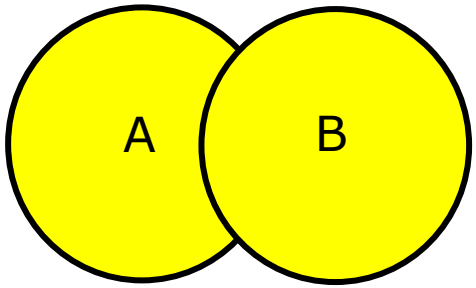
Бертран Рассел - британский философ, логик, математик, социолог, общественный деятель:
«Множество есть совокупность различных элементов, мыслимая как единое целое».

Применительно к БД результат любого запроса можно принять за множество.

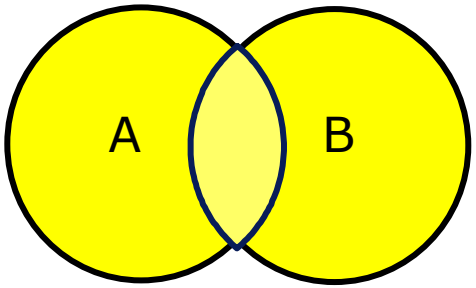
Типы операторов множеств

Операторы множеств объединяют результаты двух или более запросов в один результат.

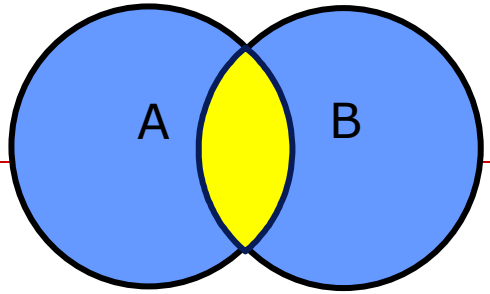
- Оператор UNION объединяет все уникальные строки.



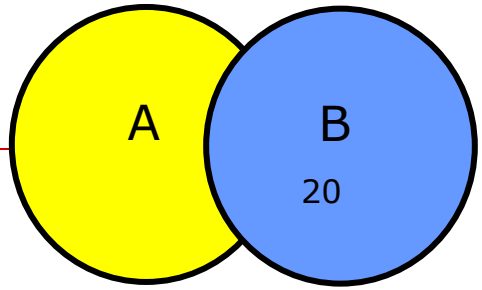
- Оператор UNION ALL объединяет все строки, включая дубликаты.



- Оператор INTERSECT объединяет все строки, которые являются общими для двух запросов.



- Оператор MINUS возвращает все уникальные строки, которые выбираются первым запросом, но не входят во второй.



Правила применения операторов множеств

- Количество столбцов объединяемых запросов, заданных в предложении SELECT, должно совпадать;
- Тип данных (групповой тип данных: NUMBER, CHAR, DATE) каждого столбца во втором запросе должен соответствовать типу данных соответствующего столбца в первом запросе;
- Допускается использование скобок для изменения последовательности действий при выполнении запроса;
- Предложение ORDER BY можно размещать только последней инструкцией оператора множеств.

Oracle Server и операторы множеств

- Дублирующие строки автоматически исключаются из результата, кроме оператора UNION ALL.
- Имена столбцов результата формируются из имен столбцов, заданных в первом операторе SELECT.
- Полученный результат по умолчанию сортируется по возрастанию значений первого столбца, кроме оператора UNION ALL.

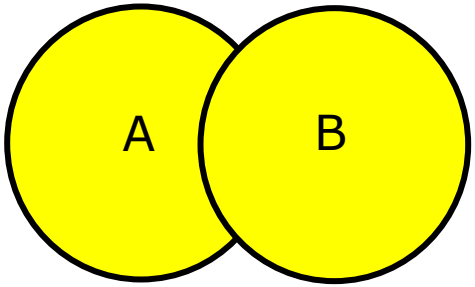
Oracle Server и операторы множеств

Результаты запросов, используемые в операторах множеств, по количеству столбцов и их типов данных (по группам) должны совпадать:

- Если оба запроса выбирают значения типа CHAR, равные по длине, то результат имеет тип CHAR данной длины;
- Если запросы выбирают значения типа CHAR с различными длинами, то результат является типом VARCHAR2 с длиной, соответствующей большему из значений CHAR;
- Если запрос/запросы выбирают значения типа VARCHAR2, то результат имеет тип VARCHAR2;
- Если запросы выбирают числовые данные, то результирующий тип данных является числовым;
- В запросах, использующих операторы множеств, сервер Oracle не выполняет неявное преобразование между разными группами типов данных, поэтому **если результаты объединяемых столбцов имеют различные типы**, то сервер **Oracle возвращает ошибку**.

Оператор UNION

Оператор UNION возвращает все строки, которые выбираются из запросов, предварительно устранив любые повторяющиеся строки.



Необходимо соблюдать следующие правила использования оператора UNION:

- Число столбцов, выбираемых запросами, и их типы данных должны совпадать;
- Наименования столбцов НЕ обязательно должны быть идентичными;
- Объединение применяется ко всем выбираемым столбцам;
- NULL-значения не игнорируются во время проверки дубликатов;
- Оператор IN имеет больший приоритет, чем оператор UNION;
- По умолчанию результат сортируется по значению первого столбца из оператора SELECT.

Использование оператора UNION

- Запрос формирует текущую и предыдущую должность сотрудников. Результат не содержит дублирующих записей:

```
SELECT employee_id,
job_id
FROM   employees
UNION
SELECT employee_id,
job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	100	AD_PRES
2	101	AC_ACCOUNT
...		
22	200	AC_ACCOUNT
23	200	AD_ASST
24	201	MK_MAN
...		

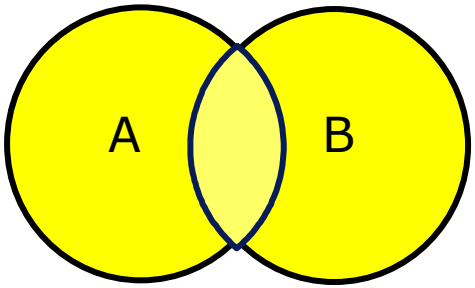
- Оператор UNION формирует 3 записи о сотруднике с EMPLOYEE_ID = 200, т.к. запросы формируют информацию о департаменте, в котором работают сотрудники:

```
SELECT  employee_id, job_id, department_id
FROM    employees
UNION
SELECT  employee_id, job_id, department_id
FROM    job_history;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
2	101	AC_ACCOUNT	110
...			
107	200	AC_ACCOUNT	90
108	200	AD_ASST	10
109	200	AD_ASST	90
...			

Оператор UNION ALL

Оператор UNION ALL возвращает строки из всех запросов, объединенных этим оператором, включая дублирующие строки.



- Для оператора UNION ALL справедливы такие же правила как для UNION, однако результат содержит дублирующие строки и не является отсортированным.
- Применение оператора DISTINCT не позволяет избавиться от дублирующих строк, т.к. они формируются в результате объединения результатов, а не в результате выборки оператором SELECT.
- Оператор UNION ALL производительнее UNION, так как не выполняются дополнительные проверки на уникальность.

Использование оператора UNION ALL

- Запрос формирует текущую и предыдущую должности сотрудников, включая дублирующие записи:

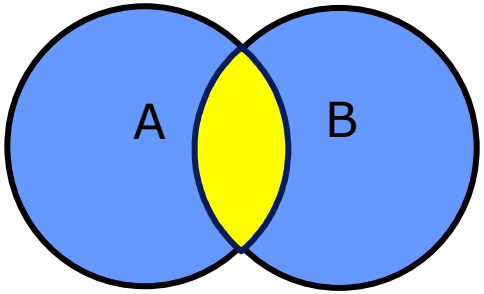
```
SELECT employee_id, job_id,
department_id
FROM   employees
UNION ALL
SELECT employee_id, job_id,
department_id
FROM   job_history
ORDER BY employee_id;
```

	EMPLOYEE_ID	JOB_ID	DEPARTMENT_ID
1	100	AD_PRES	90
...			
16	144	ST_CLERK	50
17	149	SA_MAN	80
18	174	SA_REP	80
19	176	SA_REP	80
20	176	SA_MAN	80
21	176	SA_REP	80
22	178	SA_REP	(null)
...			
30	206	AC_ACCOUNT	110

Так как оператор –UNION и UNION ALL ассоциативны и коммутативны, то **порядок объединения** более двух запросов **на результат не будет влиять**, однако **будет влиять на производительность**. Поэтому рекомендуется объединять сначала небольшие множества, а затем полученный результат объединять с большим множеством.

Оператор INTERSECT

Оператор INTERSECT возвращает только общие строки для всех запросов, к которым он применяется.



Необходимо соблюдать следующие правила использования оператора INTERSECT:

- Число столбцов, выбираемых в предложении SELECT, и их типы данных должны совпадать;
- Наименования столбцов НЕ обязательно должны быть идентичными;
- Пересечение (INTERSECT) не игнорирует NULL-значения ;
- Порядок применения оператора INTERSECT к трем и более запросам не изменяет результат.

Использование оператора INTERSECT

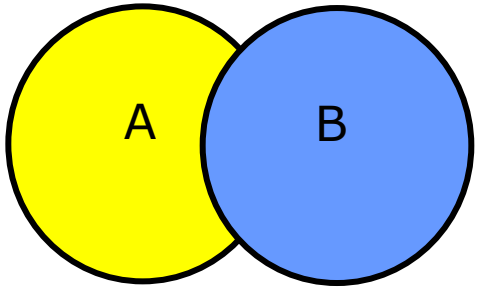
- Результат выполнения данного оператора будет состоять из ID сотрудника и его должности только для тех сотрудников, которые меняли должность в компании, а затем вернулись на прежнюю.

```
SELECT employee_id, job_id
FROM   employees
INTERSECT
SELECT employee_id, job_id
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID
1	176	SA_REP
2	200	AD_ASST

Оператор MINUS

Оператор MINUS возвращает уникальные значения из результата первого оператора SELECT, отсутствующие в результате второго оператора SELECT.



- Оператор MINUS также как и UNION обладает ограничением на результат запросов, которые должны быть совместимы по объединению, т.е. содержать одинаковое количество столбцов, и каждый столбец первого запроса должен быть того же типа данных (или автоматически приводиться к нему), что и находящийся в том же месте столбец второго запроса.

Использование оператора MINUS

- Запрос позволяет получить список сотрудников, которые никогда не меняли должность.
- В результате отсутствуют сотрудники с ID = 101 и ID = 102.

```
SELECT employee_id
FROM   employees
MINUS
SELECT employee_id
FROM   job_history;
```

	EMPLOYEE_ID
1	100
2	103
3	104
4	107
5	124
...	
14	205
15	206

Оператор SELECT и операторы множеств 1

- Данный оператор объединения формирует результат, состоящий из ID местоположения отдела, его названия и области/региона.
- Выбираемые столбцы в операторах SELECT должны соответствовать по типу данных и количеству, поэтому при отсутствии столбцов необходимо использовать функции явного преобразования (TO_CHAR, TO_DATE, TO_NUMBER) со значением NULL.

```
SELECT location_id, department_name "Department",
   TO_CHAR(NULL) "Warehouse location"
FROM departments
UNION
SELECT location_id, TO_CHAR(NULL) "Department",
   state_province
FROM locations;
```

Оператор SELECT и операторы множеств 2

- В результате запроса формируется список сотрудников (ID) с окладами и занимаемыми должностями.
- В таблице job_history отсутствует столбец salary, поэтому он заменен значением 0.

```
SELECT employee_id,
job_id,salary
FROM   employees
UNION
SELECT employee_id, job_id,0
FROM   job_history;
```

	EMPLOYEE_ID	JOB_ID	SALARY
1	100	AD_PRES	24000
2	101	AC_ACCOUNT	0
3	101	AC_MGR	0
4	101	AD_VP	17000
5	102	AD_VP	17000

...

29	205	AC_MGR	12000
30	206	AC_ACCOUNT	8300

Использование ORDER BY с операторами множеств

- Предложение ORDER BY можно использовать только один раз, последней инструкцией составного запроса;
- Предложение ORDER BY может содержать наименование столбца, псевдоним или позиционный номер столбца, объявленного в первом операторе SELECT;
- Если предложение ORDER BY не указано, то результат сортируется в порядке возрастания значений из первого столбца первого запроса.

```
SELECT employee_id, job_id,salary
FROM   employees
UNION
SELECT employee_id, job_id,0
FROM   job_history
ORDER BY 2;
```

```
SELECT employee_id, job_id,salary
FROM   employees
UNION
SELECT employee_id e, job_id j,0
n
FROM   job_history
ORDER BY job_id;
```