



Подзапросы

База данных, используемая в примерах



Задание на объединение:

**вывести все путевки туриста Иванова
(ПунктНазначения, Фамилия).**



Подзапрос – это оператор **SELECT**,
вложенный в:

- 1) предложение **WHERE** или **HAVING** другого оператора **SELECT**;
- 2) оператор **INSERT**, **UPDATE** или **DELETE**;
- 3) другой подзапрос.

Подзапрос



Некоррелированный

(не зависит от внешнего запроса)

Коррелированный

(зависит от внешнего запроса)

Пример:

Вывести сумму, которую заплатил за поездку турист Иванов

Некоррелированный подзапрос:

```
SELECT Сумма
```

```
FROM Оплата
```

```
WHERE КодТуриста IN
```

```
(SELECT КодТуриста
```

```
FROM Туристы
```

```
WHERE Фамилия = 'Иванов');
```

Коррелированный подзапрос:

```
SELECT Сумма
```

```
FROM Оплата
```

```
WHERE 'Иванов' IN
```

```
(SELECT Фамилия
```

```
FROM Туристы
```

```
WHERE Оплата.КодТуриста =  
Туристы.КодТуриста);
```

Большинство подзапросов могут
быть заменены запросом на
объединение таблиц


Запрос на объединение таблиц:

```
SELECT Сумма
```

```
FROM Оплата, Туристы
```

```
WHERE Фамилия = 'Иванов' ;
```

Ошибка?



Некоторые выборки гораздо удобнее представлять в виде подзапросов, чем в виде объединения, например, при необходимости самообъединения таблиц.

Пример:

Вывести всех туристов (фамилия, имя и отчество), телефоны которых совпадают с телефоном Журавлева Юрия Петровича

Самообъединение:

```
SELECT тур1.Фамилия, тур1.Имя,  
тур1.Отчество
```

```
FROM Туристы тур1, Туристы тур2
```

```
WHERE тур1.Телефон = тур2.
```

```
Телефон and тур2.Фамилия =  
'Журавлев' and тур2.Имя = 'Юрий'  
and тур2.Отчество = 'Петрович';
```

Подзапрос (какой?некорр/корр):

```
SELECT Фамилия, Имя, Отчество
```

```
FROM Туристы
```

```
WHERE Телефон IN
```

```
(SELECT Телефон
```

```
FROM Туристы
```


```
WHERE Фамилия = 'Журавлев'  
and Имя = 'Юрий' and Отчество =  
'Петрович');
```



Обычно :

Подзапросы используются, когда необходимо сравнивать значения агрегирующей функции с другими значениями.

Объединения используются, когда отображается информация из нескольких таблиц



Для написания подзапроса
используются следующие
операторы:

- 1) IN (или NOT IN);
- 2) операторы сравнения с
использованием или без
использования ANY или ALL ;
- 3) EXISTS (или NOT EXISTS).

ANY и ALL

- > ALL означает больше самого большого
- > ANY - больше хотя бы одного из значений
- < ALL – меньше самого меньшего
- < ANY – меньше хотя бы одного из значений
- = ANY – равно одному из значений (аналогичен оператору IN)

Пример : Кто из туристов заплатил за
платеж больше, чем любой из
Ивановых

```
SELECT Фамилия, Имя, Отчество
```

```
FROM Оплата, Туристы
```

```
WHERE Оплата.КодТуриста = Туристы.
```

```
КодТуриста and
```

```
Сумма > ALL
```

```
(SELECT Сумма
```

```
FROM Оплата, Туристы
```

```
WHERE Оплата.КодТуриста = Туристы.
```

```
КодТуриста and Фамилия = 'Иванов');
```


При использовании операторов сравнения без ANY или ALL необходимо, чтобы подзапрос возвращал только единственное значение. Например, следующий запрос этого не гарантирует:

```
SELECT Фамилия, Имя, Отчество
FROM Туристы
WHERE Телефон =
    (SELECT Телефон
     FROM Туристы
     WHERE Фамилия = 'Журавлев');
```

Исправить!

Гарантии выборки единственного значения может дать применение агрегирующих функций, например вывести коды туристов, которые заплатили самую большую сумму за путевку можно так:

```
SELECT КодТуриста
FROM Оплата
WHERE Сумма =
      (SELECT max(Сумма)
       FROM Оплата);
```

Кроме того, при использовании операторов сравнения (с использованием или без использования ALL или ANY) нужно гарантировать, что в подзапросе не будет нулевых значений, т.к. их нельзя сравнивать с другими значениями.

EXISTS

Это запросы, выполняющие проверку на существование.

Например запрос «Вывести фамилию, имя и отчество туристов, если среди них есть турист Иванов» можно выполнить следующим образом:

```
SELECT Фамилия, Имя, Отчество
FROM Туристы
WHERE EXISTS
  (SELECT *
   FROM Оплата
   WHERE Туристы.Фамилия =
    'Иванов');
```

Чего в этом запросе не хватает?

Или запрос «Вывести фамилию, имя и отчество туристов, если среди них нет должников» можно выполнить следующим образом:

```
SELECT Фамилия, Имя, Отчество
FROM Туристы
WHERE NOT EXISTS
    (SELECT *
     FROM Оплата
     WHERE Сумма == 0 or Сумма == NULL);
```

А что в этом запросе неправильно?



Подзапросы с разным уровнем вложения

Пример: Вывести фамилии туристов, которые отправились в Париж.

```
SELECT Фамилия
FROM Туристы
WHERE КодТуриста IN
    (SELECT КодТуриста
     FROM Оплата
     WHERE КодПутевки IN
        (SELECT КодПутевки
         FROM Путевка
         WHERE ПунктНазначения =
        'Париж')));
```

Подзапросы в операторе UPDATE

Пример: Уменьшить всем Ивановым сумму, оплаченную за путевку в 2 раза

```
UPDATE Оплата
SET Сумма = Сумма / 2
WHERE КодТуриста IN
(SELECT КодТуриста
FROM Туристы
WHERE Фамилия = 'Иванов');
```

Подзапросы в операторе DELETE

Пример: Удалить все оплаты за путевку в Египет

```
DELETE Оплата
```

```
WHERE КодПутевки IN
```

```
(SELECT КодПутевки
```

```
FROM Путевка
```

```
WHERE ПунктНазначения =  
'Египет');
```