Урок №5

Теория баз данных

Содержание

Подзапросы

Виды и применение вложенных подзапросов

Стандартом SQL предусмотрена возможность вкладывать запросы друг в друга, что имеет большое практическое применение. В результате такого вложения, одни запросы могут управлять другими. При этом вводится понятие подзапроса. Подзапрос — это запрос, который содержится в другом запросе SQL. Он представляет собой полноценное **SELECT** выражение, результат выполнения которого используется в другом запросе. Размещаться они могут почти в любом месте SQL выражения, например, вместо одного из имен в списке **SELECT**, в операторе **FROM**, при указании условия в операторах **WHERE** или **HAVING**. Чаще всего встречается использование подзапросов при указании условия. Следует отметить, что сами по себе подзапросы не добавляют никаких функциональных возможностей, но иногда с ними запросы становятся более читабельными, чем при сложной выборке.

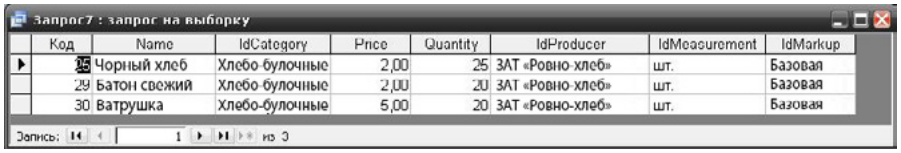
При использовании подзапросов часто используют понятие ***внешнего*** и ***внутреннего запроса***. Сначала выполняется подзапрос, то есть внутренний запрос, который размещается, например, в инструкции **WHERE**, а затем основной, то есть внешний запрос, который может

быть инструкцией **SELECT**, **INSERT**, **DELETE** или **UPDATE**. При этом ***вложенный подзапрос всегда ограничивается скобками.***

Чтобы лучше понять, как пользоваться подзапросами и как они работают, рассмотрим все на примере. Для начала, напишем запрос без использования подзапросов, который выводит всю информацию о товарах только одного поставщика. При обычных условиях мы напишем:

```
SELECT Product.*
FROM   Product, Producer
WHERE Product.IdProducer=Producer.IdProducer AND
        Producer.Name='ЗАТ "Ровно-Хлеб"';
```

Результат:

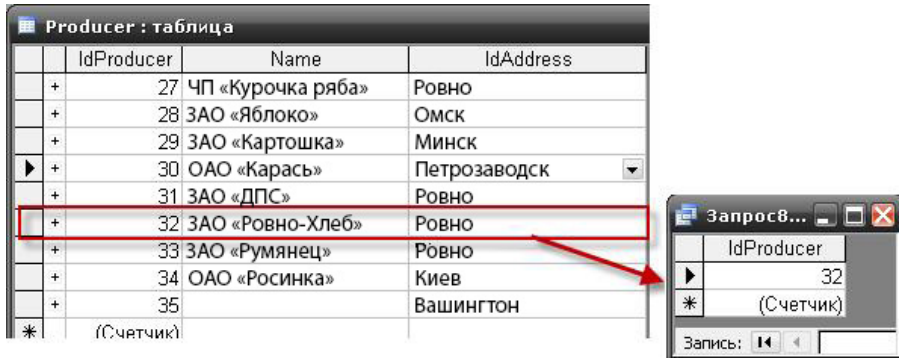


При использовании подзапросов, наш запрос перепишется следующим образом:

```
SELECT *
FROM Product
WHERE IdProducer =
                    (SELECT IdProducer
                     FROM Producer
                     WHERE Name='ЗАО "Ровно-Хлеб"');
```

Результат будет аналогичный. Что же происходит при использовании подзапроса?

Сначала целостно выполняется подзапрос, который размещается в инструкции **WHERE**. Результатом его выполнения будет идентификатор — значение поля Name равно ЗАО "Ровно-Хлеб".



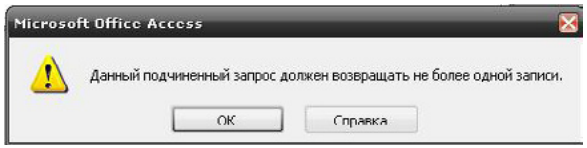
Полученный результат возвращается в основной запрос и используется при его исполнении. То есть внешние ключи, которые используются для связи с таблицей Producer, сопоставляются с первичным ключом, который является результатом выполнения внутреннего подзапроса. В результате будут выбраны только те товары, которые были изготовлены на ЗАО «Ровно-Хлеб».

Прежде, чем мы перейдем к рассмотрению других примеров по использованию подзапросов, познакомимся с ограничениями при их использовании. Они следующие:

1) ***Результатом подзапроса должно быть только одно значение, причем его тип данных должен соответствовать типу значения, которое используется во внешнем запросе.*** Например, нам нужно вывести всю информацию о товарах двух производителей: ЗАО «Ровно-Хлеб» и ОАО «Росинка».

```
SELECT *
FROM Product
WHERE Is Producer =
                    (SELECT IdProducer
                     FROM Producer
                     WHERE Name='ЗАО "Ровно-Хлеб"'
                     OR Name='ОАО "Росинка"');
```

Данный запрос приведет к ошибке, поскольку результатом данного подзапроса есть несколько значений, то есть два идентификатора, значения полей Name которых равно ЗАО "Ровно-Хлеб" и ОАО "Росинка".



Следовательно, при использовании запросов, основанных на операторах сравнения или логических операторах, следует быть очень внимательным, чтобы конечным результатом подзапроса был только одна запись.
Выходом из этой ситуации является использование оператора **IN**, который применяется для перебора значений результата работы внутреннего подзапроса:

```
SELECT *
FROM Product
WHERE Is Producer IN
                    (SELECT Id Producer
                     FROM Producer
                     WHERE Name='ЗАО "Ровно-Хлеб"'
                     OR Name='ОАО "Росинка"');
```


2. ***Результатом подзапроса может быть NULL-запись.*** В таком случае результат работы подзапроса будет оценен как **UNKNOWN**, что равносильно FALSE. В результате, предыдущий запрос будет верным даже без использования оператора **IN**, но при условии, если товаров одного из указанных производителей или обоих в базе данных не существует. В таком случае, подзапрос на выходе вернет одну или ни одной записей.

   В некоторых случаях, для подстраховки, существует также возможность использования оператора **DISTINCT** для гарантированного получения одной записи на выходе подзапроса.

3. Согласно стандарту ANSI SQL ***подзапросы является неперемещаемыми***, то есть следующий запрос является верным:

```
SELECT *
FROM Product
WHERE Is Producer =
                    (SELECT Id Producer
                     FROM Producer
                     WHERE Name='ЗАО "Ровно-Хлеб"'
                          OR Name='ОАО "Росинка"');
```

Но, если поменять тело подзапроса со сравнительным значением местами, должна сгенерироваться ошибка:

```
SELECT *
FROM Product
WHERE (SELECT IdProducer
```

```
        FROM Producer
        WHERE Name='ЗАО "Ровно-Хлеб"'
        OR Name='ОАО "Росинка"') = IdProducer;
```

ПРИМЕЧАНИЕ! *Данное правило не касается среды СУБД MS Access, поскольку она рассматривает и первый, и второй вариант как одинаковые.*

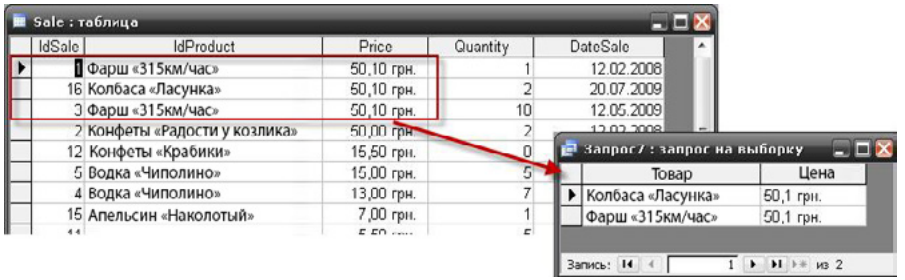
4) Вложенный запрос нельзя использовать в инструкции **ORDER BY**.
5) При использовании подзапросов для проверки результата нельзя использовать операторы **BETWEEN**, **LIKE**, **IS NULL**.
6) Если подзапрос используется с немодифицированным оператором сравнения, то есть обычным знаком равенства (**=**), без использования ключевых слов **SOME**, **ANY** или **ALL**, то он не может содержать операторы группировки **GROUP BY** и **HAVING**.

В подзапросах допускается использовать функции агрегирования, поскольку их результатом является единственное значение. Но и здесь следует быть внимательным (см. п.6 замечаний). Рассмотрим несколько примеров:

1. Запрос на получение информации о наиболее дорогом товаре, то есть товар с наибольшей ценой продажи:

```
SELECT DISTINCT Product.Name as [Товар],
                Sale.Price & 'грн.' as [Цена]
FROM Product, Sale
WHERE Product.IdProduct=Sale.IdProduct
      AND Sale.Price=(SELECT max(Sale.Price)
FROM Sale)
```

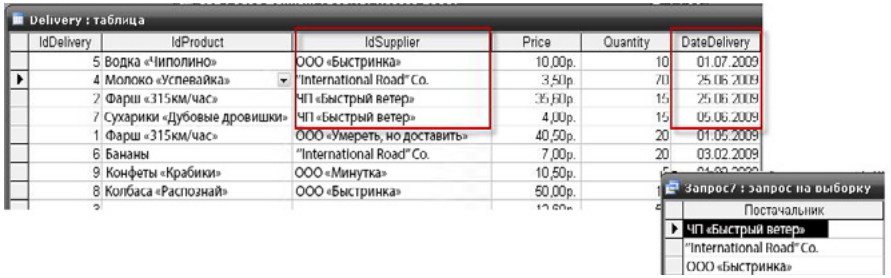

Результат:



2. Запрос, который выводит на экран имена и фамилии всех поставщиков, которые поставляли товар в промежутке между 01/06/2009 и текущей датой:

```
SELECT Name as [Поставщик]
FROM Supplier
WHERE IdSupplier IN (SELECT IdSupplier
                     FROM Delivery
                     WHERE DateDelivery
                     BETWEEN #01/06/2009# AND Date());
```

Результат:



3. Получить информацию о всех поставщиках, которые поставляли товар больше, чем 2 раза. Выборку отсортировать по названию производителя:

```
SELECT *
FROM Supplier s
WHERE 2 > ( SELECT COUNT(Delivery.IdSupplier)
            FROM Delivery
            WHERE s.IdSupplier = Delivery.IdSupplier)
ORDER BY 2;
```

4. Определить, какие поставляемые товары были произведены в г. Киев:

```
SELECT Name as [Товар], Format(Price, '## ###.00 грн.')
            as [Цена]
FROM Product
WHERE IdProduct IN ( SELECT DISTINCT IdProduct
    FROM Delivery
    WHERE IdSupplier IN (SELECT s.IdSupplier
                FROM Supplier s, Address addr
                WHERE s.IdAddress=addr.IdAddress
                        AND addr.Town = 'Київ'));
```

Как уже было сказано, подзапрос можно использовать в выражении **FROM** внешнего запроса. Это позволяет создать временную таблицу и добавить ее в запрос. Например, рассмотрим следующий простой запрос:

```
SELECT IdProduct, Name, IdCategory
FROM Product
WHERE Price BETWEEN 20 AND 50;
```

Данный запрос выведет на экран список товаров и идентификаторы их категорий, цена которых находится в пределах 20-50 грн. Этот запрос можно использовать в рамках другого, чтобы получить дополнительную

информацию. Например, используем его, чтобы вывести список товаров категорий «Мясные» и «Колбасные», цены которых находятся в вышеуказанном диапазоне.

```
SELECT pr.Name as [Товар]
FROM (SELECT IdProduct, Name, IdCategory
        FROM Product
        WHERE Price BETWEEN 20 AND 50) as pr,
        Category as c
WHERE pr.IdCategory = c.IdCategory AND c.Name IN
('Мясные', 'Колбасные');
```

Итак, мы используем подзапрос для создания таблицы, содержащей только три поля: IdProduct, Name и IdCategory. Таблице мы присваиваем псевдоним pr. После этого к созданной таблице можно обратиться с запросом, как к любой другой таблице. В данном случае, мы используем ее, чтобы получить информацию о товарах необходимых категорий.

Операторы EXIST, ANY, SOME, ALL

В предыдущем разделе мы с вами рассмотрели в основном однострочные запросы, то есть те, которые возвращают одну запись. Для того, чтобы обработать несколько записей, которые вернет подзапрос (многострочный подзапрос), мы использовали оператор **IN**. Но он не единственный. Кроме оператора **IN**, существует еще 4 логических оператора: **EXISTS**, **ALL**, **ANY** и **SOME**. Рассмотрим их и начнем с оператора **EXISTS**.

Оператор **EXISTS** используется, когда необходимо определить наличие значений, удовлетворяющих условию

в подзапросе. Если данные на выходе подзапроса существуют, тогда данный оператор вернет значение true, иначе — false. При использовании оператора **EXISTS** мы на самом деле используем в подзапросе данные внешнего запроса. Такой запрос иногда называют ***связанным*** или ***коррелированным подзапросом***.

Рассмотрим следующий запрос: вывести информацию о всех поставщиках, которые когда-либо поставляли товары в магазин:

```
SELECT *
FROM Supplier
WHERE EXISTS (SELECT *
                FROM Delivery
                WHERE Supplier.IdSupplier=
                                        Delivery.IdSupplier)
ORDER BY 3;
```

Проанализируем, что получилось в результате. В подзапросе мы ищем записи таблицы Delivery, в которых значение IdSupplier совпадает со значением Supplier.IdSupplier. Каждая запись таблицы Supplier сопоставляется с результатом подзапроса и ***В СЛУЧАЕ СУЩЕСТВОВАНИЯ*** (**WHERE EXISTS**) информация о поставщике добавляется в результирующую множество.

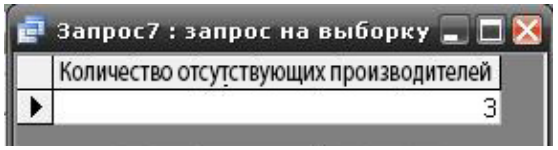
Кстати, то, что возвращается подзапросом в подзапрос **EXISTS** или **NOT EXISTS** абсолютно не имеет значения. В связи с этим, иногда возвращают произвольное константное значение. Все, что необходимо знать, — это сам факт наличия или отсутствия записей, удовлетворяющих критерию подзапросов. К примеру:


```
SELECT *
FROM Supplier
WHERE EXISTS (SELECT 'X'
                FROM Delivery
                WHERE Supplier.IdSupplier =
                                        Delivery.IdSupplier)
ORDER BY 3;
```

Приведем еще один пример по использованию данного оператора. Необходимо вывести информацию о количестве производителей, информация о которых есть в базе данных, но товаров которых в магазине нет.

```
SELECT COUNT(*) as [Количество отсутствующих производителей]
FROM Producer
WHERE NOT EXISTS ( SELECT *
                    FROM Product
                    WHERE Product.IdProducer =
                            Producer.IdProducer);
```

Результат:



Операторы **ALL**, **ANY** и **SOME** используются для сравнения одного значения с множеством данных, которые возвращаются подзапросом. Их можно комбинировать со всеми операторами сравнения и они могут включать инструкции **GROUP BY** и **HAVING**. Стоит отметить, что

операторы **ANY** и **SOME** вообще идентичны и в стандарте ISO указывается, что они эквивалентны.

Рассмотрим для начала действия оператора **ANY**.

- **= ANY** — равно произвольному значению, которое возвращается под запросом. Приравнивается к действию оператора **IN** и может быть им заменено
- **<ANY** — меньше наибольшего значения, которое возвращается подзапросом. Это можно трактовать как «меньше любого значения».
- **> ANY** — болеше наименьшего значения, которое возвращается подзапросом. Это можно трактовать как «больше любого значения».

Для демонстрации работы данного оператора перепишем запрос на получение информации о поставщиках, которые поставляли товары в магазин с использованием оператора **ANY**:

```
SELECT IdSupplier, Name
FROM Supplier
WHERE IdSupplier = ANY ( SELECT IdSupplier
                         FROM Delivery)
ORDER BY 2;
```

В данном случае, оператор **ANY** сопоставляет все значения поля IdSupplier из таблицы Delivery и возвращает результат true, если ***ЛЮБОЕ*** (**ANY**) значение совпадет со значением поля IdSupplier.

Что же до оператора **SOME** (*какой-нибудь*), то он даст аналогичный результат.


```
SELECT IdSupplier, Name
FROM Supplier
WHERE IdSupplier = SOME ( SELECT IdSupplier
                          FROM Delivery)
ORDER BY 2;
```

Разница между операторами **ANY** и **SOME** заключается лишь в терминологии и заключается в том, чтобы позволить людям использовать тот термин, который наиболее подходит в данной ситуации.

Использование оператора **ALL** также не сложно. Данный оператор возвращает значение true, если каждое значение, которое будет получено в результате работы подзапроса, соответствует условию внешнего запроса. Принцип его действия отражен в следующей таблице:

- **>ALL** больше наибольшего значения, которое возвращается подзапросом. Это можно трактовать как «больше всех значений».
- **<ALL** — меньше наименьшего значения, которое возвращается подзапросом. Это можно трактовать как «меньше всей значений».

В качестве примера выполним запрос, в котором поставим целью доказать, что товары категории "Фрукты" является наиболее продаваемыми.

```
SELECT Product.Name
FROM Product, Sale
WHERE Product.IdProduct = Sale.IdProduct
        AND Sale.Quantity > ALL ( SELECT Sale.Quantity
                                  FROM Sale, Product, Category
```

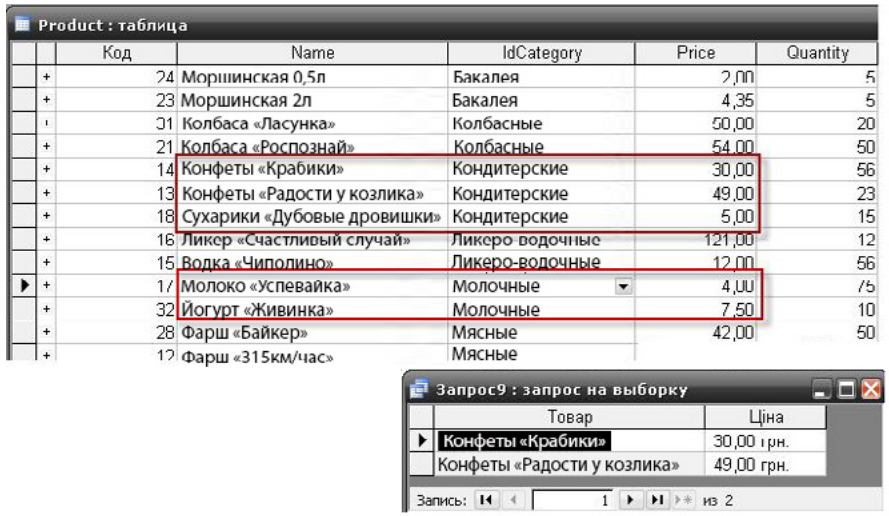
```
WHERE Sale.IdProduct=Product.IdProduct
AND Product.IdCategory=Category.IdCategory
AND Category.Name='Фрукты');
```

Подзапрос вернет список значений количества продаж (Sale.Quantity) товаров категории "Фрукты". Затем внешний запрос ищет количество продаж товаров, которые были больше, чем данные, используя для этого выражение **Sale.Qauntity> ALL** (*количество продаж*). Если данный запрос не вернет ни одной записи — это будет подтверждением того, что товары данной категории действительно наиболее продаваемые, иначе выведется перечень товаров, которые продаются более ниш указанной в подзапросе категории.

Еще один пример. Найдем и выведем названия и цены кондитерских изделий, стоимость которых превышает стоимость товаров категории «Молочные продукты».

```
SELECT DISTINCT Product.Name as [Товар],
Format(Product.Price, '## ###.00 грн.') as [Цена]
FROM Product, Category
WHERE Product.Price > ALL (SELECT Product.Price
        FROM Product, Category
        WHERE Product.IdCategory=Category.IdCategory
        AND Category.Name='Молочные')
        AND Product.IdCategory=Category.IdCategory
        AND Category.Name='Кондитерські';
```

Результат:



Объединение запросов.

Операторы UNION и UNION ALL

Объединение — это связывание данных, содержащихся в двух таблицах или запросах в один результирующий набор. Поскольку объединение запросов и таблиц отличается, рассмотрим отдельно один и второй способ объединения. Начнем с простого, а именно с объединения результатов двух или более запросов.

Объединение запросов осуществляется с помощью оператора SQL **UNION**. С его помощью можно объединить результаты от 2 до 255 результатов запросов в один результирующий набор. В результате такого объединения одинаковые записи по умолчанию уничтожаются, но при наличии ключевого слова **ALL** (т.е. при использовании оператора **UNION ALL**) возвращаются все записи, в том числе и одинаковые. Синтаксис оператора **UNION** следующий:

```
SELECT <список_полей>
[FROM <список_таблиц>]
[WHERE <условие>]
[GROUP BY <список_полев_для_группирования>]
[HAVING <условие_на_группу>]
UNION [ALL]
SELECT <список_полев>
[FROM <список_таблиц>]
[WHERE <условие>]
[GROUP BY <список_полей_для_группирования>]
```



```
[HAVING <условие_на_группу>];
[ORDER BY <условие_сортировки>];
```

При использовании оператора **UNION** ***придерживаются следующих правил***:

- количество, последовательность и типы данных полей в обоих списках **SELECT** должны совпадать;
- операторы **GROUP BY** и **HAVING** используются только в одном запросе;
- ни одна из таблиц не может быть отсортирована отдельно; можно сортировать только результирующий запрос. Поэтому оператор **ORDER BY** можно использовать только в конце оператора **UNION**;
- имена полей результирующей выборки определяются списком полей первого оператора **SELECT**.

Для начала выберем все товары, цена которых больше 20 грн., ИЛИ код категории товара равен 1.

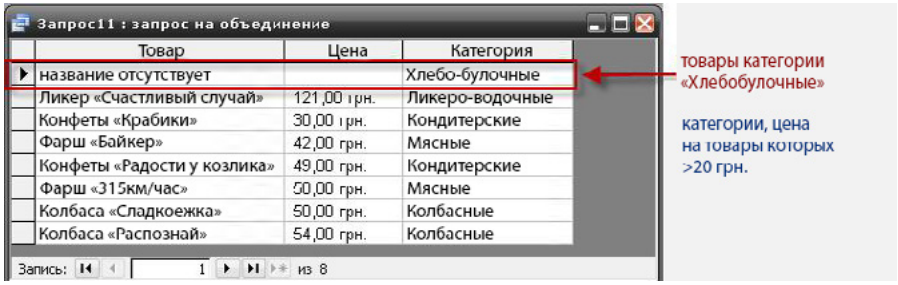
```
SELECT Name
FROM Product
WHERE Price > 20
UNION
SELECT Name
FROM Product
WHERE IdCategory=1;
```

Результат:

А теперь выберем все товары, цена которых больше 20 грн., ***ИЛИ*** которые относятся к категории "Хлебо-булочные".

```
SELECT pr.Name as [Товар],
       Format(pr.Price, '## ###.00 грн.') as [Цена],
       c.Name as [Категория]
FROM Product pr, Category c
WHERE  pr.IdCategory = c.IdCategory AND pr.Price > 20
UNION
SELECT 'наименование отсутствует', NULL, Name
FROM Category
WHERE Name ='Хлебо-булочные';
```

Результат:



Объединение таблиц.

Стандарт SQL2. Виды объединений

Виды объединений

Одной из самых сильных сторон SQL является возможность связывать данные, размещаемые в отдельных таблицах. Это связывание осуществляется за счет объединения таблиц, которые указываются в списке **FROM**. Вместе с этим задается способ или тип объединения.

СУБД MS Access поддерживает только три типа объединений:

1) внутреннее, которое иногда называют простым;
2) внешнее;
3) самообъединение.

Другие СУБД, кроме вышеперечисленных, поддерживают и другие типы объединений. Причем, некоторые специфические только для них. Но эти три типа объединений являются основными и поддерживаются всеми.

Для настройки объединения, следует воспользоваться одним из синтаксисов стандарта ANSI: старого стандарта SQL'86 или стандартов SQL2 и выше.

Они выглядят следующим образом:

```
-- SQL'86
SELECT [таблица.] Поле [... n]
FROM таблица [таблица] [...]
WHERE  условие_объединения
```

```
- -SQL2 и выше
SELECT [таблица.] Поле [... n]
FROM таблица [тип_ объединения] JOIN таблица
ON условие_объединения
```

Как видно из описания, синтаксис стандартов отличается. Причем, видим, что к этому времени мы пользовались старым стандартом ANSI SQL, в котором нет возможности указывать тип объединения.

Согласно новому стандарту тип объединения указывается ключевым словом при связи двух таблиц. Условие объединения, которое теперь указывается после оператора **ON**, представляет собой выражение, аналогичное условию отбора, который использовался в старом стандарте в выражении **WHERE**. Она задает, как будут относиться между собой записи в двух таблицах. Большинство операций связывания выполняются на основе выражений эквивалентности, таких как ПолеА = ПолеВ. Однако, условие объединения может быть и больше, при этом все выражения, которые входят в условие объединяются с помощью логических операторов **AND** или **OR**. В этом плане ничего не изменилось.

Внутренние объединения. Оператор INNER JOIN

Первый вид объединения, который мы рассмотрим, будет внутреннее объединение, которое представляет собой обычное объединение двух или более таблиц. На самом деле, вы его использовали раньше. Старый стандарт ANSI SQL использует внутренний тип объединения для связи таблиц.

В стандартах SQL2 и выше, внутреннее объединение осуществляется с помощью оператора **INNER JOIN**. Причем использование данного оператора без ключевого слова **INNER** также допускается (СУБД MS Access исключение). В результате такого объединения получается новая таблица, записи которой удовлетворяют соответствующим условиям. Как вы уже поняли, внутренние объединения возвращают данные, если находят общую информацию в обеих таблицах.

Например, отобразим в очередной раз информацию о товарах и их категории.

```
SELECT pr.Name as [Товар], c.Name as [Категория]
FROM Product pr, Category c
WHERE pr.IdCategory = c.IdCategory;
```

Но данная запись использует старый стандарт ANSI SQL. Перепишем его в соответствии с требованиями стандартов ANSI SQL2 и выше, то есть с использованием оператора **INNER JOIN**.

```
SELECT pr.Name as [Товар], c.Name as [Категория]
FROM Product pr INNER JOIN Category c
ON pr.IdCategory = c.IdCategory;
```

Результат будет одинаков. Фактически, отличие заключается лишь в том, что связи между таблицами указываются с помощью оператора **INNER JOIN**, а связи по ключевым полям описываются после оператора **ON**. Все остальные операторы действуют так же, как и раньше.

Рассмотрим еще один пример, в котором в связи будет принимать участие более двух таблиц. Добавим к нашему запросу информацию о производителе товара.

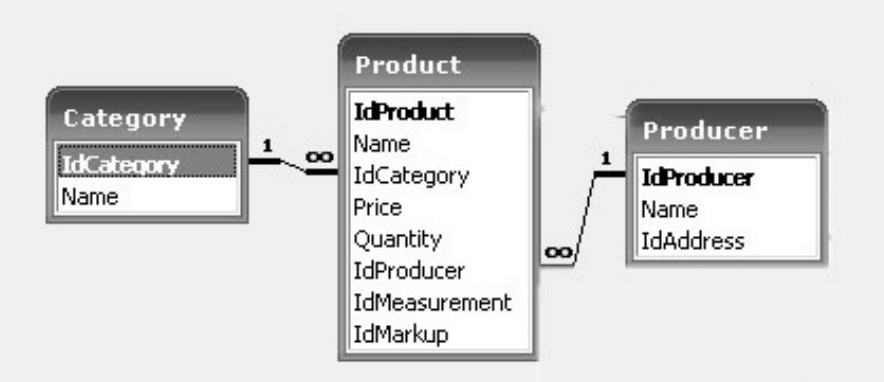
Согласно старому стандарту ANSI SQL, такой запрос будет выглядеть так:

```
SELECT Product.Name as [Товар],
       Category.Name as [Категория],
       Producer.Name as [Производитель]
FROM Product, Category, Producer
WHERE Product.IdProducer=Producer.IdProducer
       AND Product.IdCategory = Category.IdCategory;
```

Согласно стандарту SQL 2 и выше:

```
SELECT Product.Name as [Товар],
       Category.Name as [Категория],
       Producer.Name as [Производитель]
FROM Category INNER JOIN (Product INNER JOIN
     Producer ON Product.IdProducer = Producer.IdProducer)
          ON Product.IdCategory=Category.IdCategory;
```

Следовательно, при использовании нового стандарта ANSI SQL2 следует только просто запомнить один принцип: ***при связи таблиц, они должны образовывать сплошную последовательную цепь***. В нашем случае она следующая:



По поводу наложения условия, то здесь ничего не изменилось, условие должно размещаться в инструкции **WHERE**. Итак, наложим условие на вывод товаров только категории "Бакалея":

```
SELECT Product.Name as [Товар],
       Category.Name as [Категория],
       Producer.Name as [Производитель]
FROM Category INNER JOIN (Product INNER JOIN
                          Producer ON Product.IdProducer=
                          Producer.IdProducer)
                          ON Product.IdCategory=
                          Category.IdCategory
WHERE Category.Name='Бакалія';
```

Внешние объединения (OUTER JOIN) и их типы: левое, правое и полное

При использовании оператора **INNER JOIN** СУБД ищет и выводит записи, которые удовлетворяют критериям поиска для двух или нескольких таблиц. Но что делать в случаях, когда необходимо найти записи одной таблицы, соответствий которых нет в другой?

Например, нужно отобразить информацию обо всех товарах и поставщиках, которые их поставляли.

```
SELECT Supplier.Name AS Поставщик, Product.Name
FROM Supplier INNER JOIN (Product INNER JOIN
                          Delivery ON Product.IdProduct =
                          Delivery.IdProduct) ON Supplier.
                          IdSupplier = Delivery.IdSupplier;
```

Результат:

Запрос10 : запрос на выборку

Поставщик	Name
ООО «Умереть, но доставить»	Фарш «315км/час»
ЧП «Быстрый ветер»	Фарш «315км/час»
ЧП «Быстрый ветер»	Водка «Чиполино»
"International Road" Co.	Молоко «Успевайка»
ООО «Быстринка»	Водка «Чиполино»
"International Road" Co.	Бананы
ЧП «Быстрый ветер»	Сухарики «Дубовые дровишки»
ООО «Быстринка»	Колбаса «Распознай»
ЗАО «Минутка»	Конфеты «Крабики»
ООО «Крещатик»	Картофель «Зеленое Чудо»
ООО «Крещатик»	Конфеты «Крабики»

Как видно из результатов запроса на экране отразились только те товары, которые поставлялись и о которых существует информация в базе данных. Чтобы ***отобразить поставщиков, информация о которых присутствует в базе данных, независимо от того, поставлял он уже какой-то товар в магазин или нет***, нужно воспользоваться внешним объединением таблиц.

Внешние объединения осуществляются с помощью оператора **OUTER JOIN** и используются в случае, когда нужно, чтобы запрос возвращал все записи из одной или более таблиц, независимо от того, имеют ли они соответствующие записи в другой таблице. Фактически, они позволяют ограничить количество возвращаемых полей одной таблицы, не ограничивая при этом их для другой таблицы.

Стандарт ANSI выделяет следующие ***типы внешних объединений*** и операторы, которые осуществляют:

1. **LEFT OUTER JOIN**. ***Левое объединение*** — записи первой таблицы (*слева*) включаются в результирующую таблицу полностью, а из другой таблицы (*справа*) в результат включаются только те, которые имеют пару в первой таблице. В качестве пары для записей первой таблицы, которые не имеют пары в другой, используют пустые (*NULL*) поля.
2. **RIGHT OUTER JOIN**. ***Правое объединение*** — наоборот.
3. **FULL OUTER JOIN**. ***Полное объединение*** — включает все сопоставляемые и не сопоставляемые записи из обеих таблиц.

Но СУБД MS Access поддерживает только два первых вида внешних объединений: левое и правое.

Для демонстрации работы внешних объединений перепишем вышерассмотренный запрос таким образом, чтобы он выводил список поставщиков, информация о которых присутствует в базе данных, независимо от того, поставлял он уже какой-то товар в магазин или нет:

```
SELECT Supplier.Name, Product.Name
FROM Supplier LEFT OUTER JOIN (Delivery LEFT OUTER JOIN
        Product ON Product.IdProduct = Delivery.IdProduct)
        ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Результат:

Запрос9 : запрос на выборку

Supplier.Name	Product.Name
ЧП Вася	
ЗАО «Минутка»	Конфеты «Крабики»
ООО «Крещатик»	Картофель «Зеленое чудо»
ООО «Крещатик»	Конфеты «Крабики»
ЧП Кулаков В.В.	
ООО «Умереть или доставить»	Фарш «315кч/ч»
ЧП «Быстрый ветер»	Фарш «315кч/ч»
ЧП «Быстрый ветер»	Водка «Чиполино»

В результате запроса, если поставщик не поставлял еще товар в магазин, ему соответствует NULL-значения в поле названия поставляемого товара.

И наоборот. Если необходимо отобразить полный список товаров, с информацией об их поставщиках, независимо от того, известностна о них информация или нет:

```
SELECT Supplier.Name AS Поставщик,
       Product.Name AS Товар
FROM Supplier RIGHT OUTER JOIN (Product RIGHT OUTER JOIN
     Delivery ON Product.IdProduct = Delivery.IdProduct)
       ON Supplier.IdSupplier = Delivery.IdSupplier;
```

Такой запрос позволит сразу выявить товары, информацию о поставщиках которых не заполнили. В нашей базе данных предусмотрено, чтобы такой ситуации не случилось, ведь наличие такой информации является важным.

Самообъединение таблиц

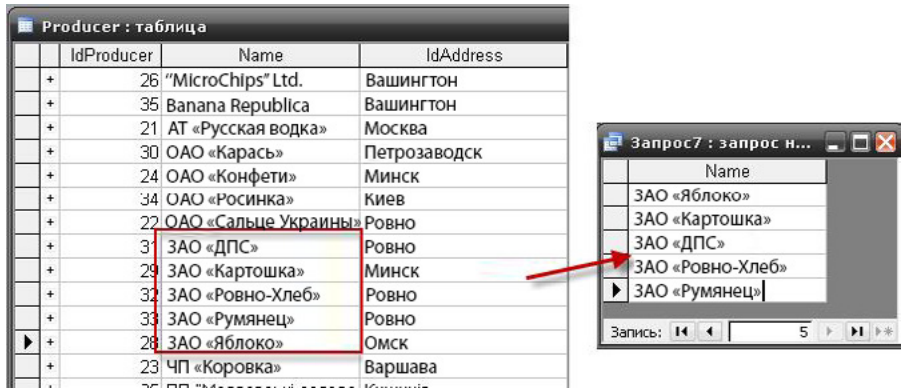
Аналогично объединению нескольких таблиц, таблицу можно объединить саму с собой. Такой вид объединения носит название самообъединение. Это может понадобиться, когда Вам будут нужны связи между строками одной и той же таблицы. Например, следующий запрос выведет информацию о производителях, названия которых начинаются с «ЗАО», то есть форма ответственности которых Закрытое Акционерное Общество:

```
SELECT p1.Name
FROM Producer p1, Producer p2
WHERE p1.IdProducer=p2.IdProducer AND p1.Name LIKE 'ЗАО*';
```


Или:

```
SELECT p1.Name
FROM Producer p1 INNER JOIN Producer p2
       ON p1.IdProducer =p2.IdProducer
WHERE p1.Name LIKE 'ЗАО*';
```

Результат:



В таком запросе для таблицы **Producer** мы определили два разных псевдонима, то есть мы сообщаем самой СУБД, что мы хотим иметь две разные таблицы, которые должны содержать одинаковые данные. После этого мы их объединяем так же, как и любые другие таблицы. А дальше получаем записи, удовлетворяющие условию.

Сначала это может показаться необычным, но при работе с несколькими таблицами в запросах идея самообъединения таблиц не должна вызывать больших трудностей.

Домашнее задание

1. Получить информацию о всех производителях, товары которых продались больше, чем 2 раза
2. Вывести самый популярный товар в магазине, то есть самый продаваемый.
3. Если общее количество товаров всех категорий считать за 100%, то необходимо подсчитать, сколько товаров каждой категории (в процентном отношении) продавалось.
4. Используя подзапросы, вывести коды и названия всех товаров, поставляемых только определенным поставщиком, например, ЗАО «Быстрый ветер».
5. Используя подзапросы вывести список товаров, их цены и категории, которые поставляются вообще только одним поставщиком.
6. Используя подзапросы, вывести названия поставщиков, которые не поставляли указанный товар (например, "Йогурт").
7. Используя подзапросы, вывести на экран список производителей, которые расположены в той же стране, что и, например, поставщик ООО Ваня.
8. Написать запрос, который выводит на экран список поставщиков одной страны (например, Украину). Использовать для вывода результата самообъединение.
9. Подсчитать количество поставщиков, товары которых поставлялись в период с 01/03/2009 по 01/06/2009 и

не были проданы. Используйте для этого оператор **EXISTS**.

10. Выведите список производителей, которые размещаются не в Украине. Отсортируйте выборку в возрастающем порядке названий производителей. Для данного подзапроса не используйте объединения таблиц, а только коррелированные подзапросы и оператор **EXISTS**.
11. Вывести на экран название товара, поставщика, который его поставлял, его полный адрес (в одном поле), категория которых «Бакалея» и «Фрукты». Учесть при выводе только те товары, которые поставляются частными предпринимателями.
12. Используя подзапросы, найти поставщиков, товаров которых нет в продаже. Используйте для поиска оператор **ANY** или **SOME**.
13. Выберите всех производителей, товаров которых в магазине в наличии больше, чем любого товара производителя ЗАО «Ласуня».
14. Вывести информацию о том, товаров каких производителей в базе данных нет. Для вывода полноценной информации воспользуйтесь внешним объединением.
15. Отразить все товары, категорий «Кондитерские» и «Хлебо-булочные» и названия поставщиков, которые их поставляли (использовать оператор **UNION**).
16. Получить информацию о количестве поставщиков двух стран (например, Украины и России), товары которых существуют в базе данных. При этом вывести отдельно полученную информацию и общую сумму

всех поставщиков товаров. Воспользуйтесь для этого операторами **UNION** и **UNION ALL**.

Требования к выполнению ДЗ. Все многотабличные запросы должны быть написаны в соответствии со стандартом SQL2.