



СУБД Microsoft Access
2003
Элементы языка SQL

Язык SQL

- SQL (Structured Query Language) – структурированный язык запросов
Язык SQL применяется во многих СУБД

Существуют стандарты языка, но его реализация отличается для различных СУБД

SQL удобен в системах «клиент-сервер»

Язык SQL

■ Применение SQL в СУБД Access

В Microsoft Access язык SQL применяется при построении запросов к базе данных



В частности, запрос на языке SQL может быть напрямую задан в полях свойств «**Источник записей**» или «**Источник строк**» элементов форм и отчетов, чтобы обойтись без создания отдельного объекта «Запрос»

Также SQL используется при работе с внешними источниками данных (например, СУБД Oracle или Microsoft SQL Server)

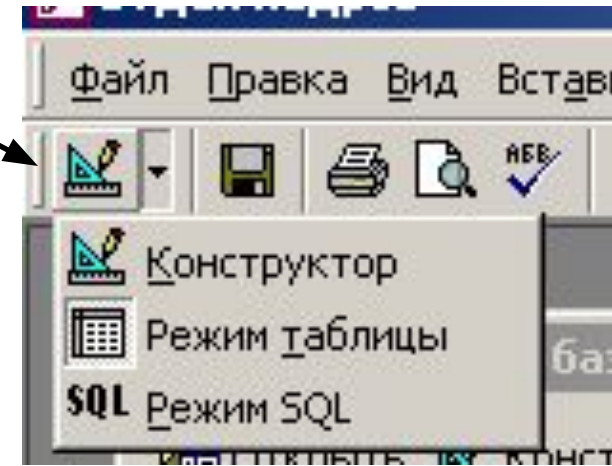
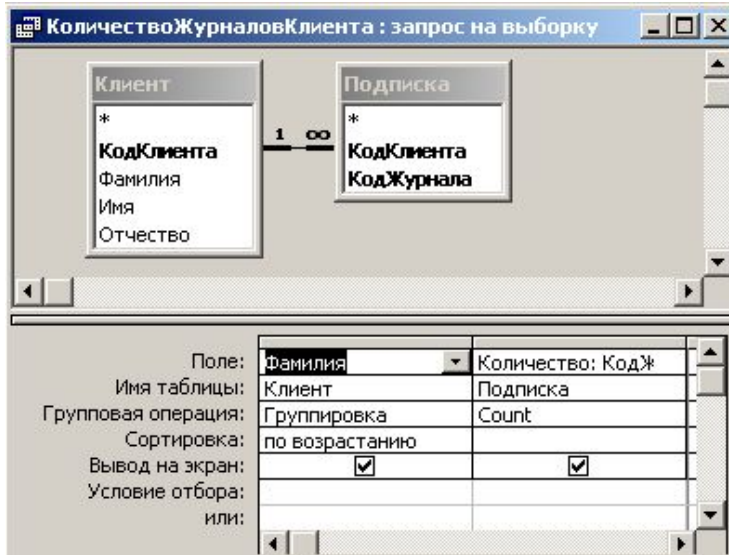
Для многих задач может оказаться достаточным использование мастера или конструктора, но не для всех



Любой запрос, построенный с помощью мастера или конструктора, может быть представлен на языке SQL. Обратное неверно.

SQL и конструктор запросов

Для переключения режимов конструктора используется кнопка «**Вид**» панели инструментов



Фамилия	Количество
Иванов	2
Николаев	1
Петров	1

Запись: 1

Оператор SELECT

При построении запросов на извлечение данных из БД в SQL используется оператор **SELECT**

Простейшая форма оператора SELECT:

SELECT <список полей>
FROM <список таблиц>;

Любой оператор SQL завершается символом «точка с запятой»

Пример 1:

SELECT Фамилия, Имя, Отчество
FROM Клиент;

Извлечь список всех клиентов (ФИО) из таблицы «Клиент»

Пример 2:

SELECT *
FROM Клиент;

Извлечь список всех клиентов (все поля) из таблицы «Клиент»

Оператор SELECT

■ Результаты выполнения запросов

Пример 1



	Фамилия	Имя	Отчество
▶	Иванов	Иван	Иванович
	Петров	Петр	Петрович
	Николаев	Николай	Николаевич
	Федоров	Федор	Федорович
*			

Запись: 1 из 4

Пример 2



	КодКлиента	Фамилия	Имя	Отчество
▶	1	Иванов	Иван	Иванович
	2	Петров	Петр	Петрович
	3	Николаев	Николай	Николаевич
	4	Федоров	Федор	Федорович
*	(Счетчик)			

Запись: 1 из 4

Оператор SELECT

Объединение нескольких таблиц в

запросе

Пример:

Есть две таблицы.
Построим запрос, выводящий фамилии клиентов и коды журналов.



```
SELECT Клиент.Фамилия, Подписка.КодЖурнала
FROM Клиент, Подписка;
```

Результат представляет собой **декартово произведение** исходных таблиц. Каждая запись таблицы «**Клиент**» объединяется с каждой записью таблицы «**Подписка**»

Следует ограничить результирующее множество записями, которые связаны между собой (содержат одинаковые значения в полях «**КодКлиента**»)

Запрос1 : запрос на выборку

	Фамилия	КодЖурнала
▶	Иванов	1
	Петров	1
	Николаев	1
	Федоров	1
	Иванов	2
	Петров	2
	Николаев	2
	Федоров	2
	Иванов	1
	Петров	1
	Николаев	1
	Федоров	1
	Иванов	3

Запись: 1 из 16

Оператор SELECT

Объединение нескольких таблиц в

В предложении «**FROM**» вместо оператора объединения «**,**» (запятая) будем использовать оператор «**INNER JOIN**» - внутреннее объединение.

Оператор «**INNER JOIN**» позволяет наложить ограничение на объединяемые записи. Предложение **ON <условие>** определяет связь между полями объединяемых таблиц

```
SELECT Клиент.Фамилия, Подписка.КодЖурнала  
FROM Клиент INNER JOIN Подписка  
    ON Клиент.КодКлиента = Подписка.КодКлиента;
```

Результат содержит только записи, для которых выполняется заданное условие

	Фамилия	КодЖурнала
▶	Иванов	1
	Иванов	3
	Петров	2
	Николаев	1
*		

Оператор SELECT

Виды объединения

➔ **INNER JOIN** (внутреннее объединение)

В результат включаются только те записи из обеих таблиц, которые связаны между собой

➔ **LEFT JOIN** (левое внешнее объединение)

В результат включаются все записи из первой таблицы. Если для них нет связанных записей во второй таблице, соответствующие поля результата будут пустыми

➔ **RIGHT JOIN** (правое внешнее объединение)

Операция, зеркально симметричная левому объединению



При создании запроса в режиме конструктора для таблиц, связи которых заданы в схеме данных, автоматически определяется операция «**INNER JOIN**»



В режиме конструктора тип объединения можно изменить заданием свойств связи

Оператор SELECT

Виды объединения

Пример

Созданный ранее запрос и результат его работы:

```
SELECT Клиент.Фамилия, Подписка.КодЖурнала  
FROM Клиент INNER JOIN Подписка  
    ON Клиент.КодКлиента = Подписка.КодКлиента;
```

	Фамилия	КодЖурнала
▶	Иванов	1
	Иванов	3
	Петров	2
	Николаев	1
*		

Тот же запрос с использованием левого объединения:

```
SELECT Клиент.Фамилия, Подписка.КодЖурнала  
FROM Клиент LEFT JOIN Подписка  
    ON Клиент.КодКлиента = Подписка.КодКлиента;
```

	Фамилия	КодЖурнала
▶	Иванов	1
	Иванов	3
	Петров	2
	Николаев	1
	Федоров	
*		

Здесь присутствует фамилия «**Федоров**», для которой нет соответствий в таблице «Журнал». Вторая колонка этой строки содержит значение «**Null**» (пусто).

Оператор SELECT

■ Группировка

Усовершенствуем составленный ранее запрос

Требуется для каждого клиента (фамилии) подсчитать количество журналов, на которые он подписан

Зададим группировку записей по фамилии (фамилии не должны повторяться) и для групп определим групповую операцию «подсчет количества»

```
SELECT Клиент.Фамилия, Count(Подписка.КодЖурнала)  
FROM Клиент LEFT JOIN Подписка  
      ON Клиент.КодКлиента = Подписка.КодКлиента  
GROUP BY Клиент.Фамилия;
```



В предложении **GROUP BY** могут быть перечислены несколько полей через запятую



Для всех полей, не вошедших в предложение **GROUP BY**, должны быть определены групповые операции

Оператор SELECT

■ Сортировка. Имена (псевдонимы)

Результат работы запроса:

Что еще осталось сделать?


То, что фамилии расположены по алфавиту – совпадение. Следует явно указать способ сортировки

Надо задать осмысленное название второго столбца (сейчас название сформировано автоматически)

	Фамилия	Expr1001
▶	Иванов	2
	Николаев	1
	Петров	1
	Федоров	0

```
SELECT Клиент.Фамилия, Count(Подписка.КодЖурнала) AS Количество  
FROM Клиент LEFT JOIN Подписка  
    ON Клиент.КодКлиента = Подписка.КодКлиента  
GROUP BY Клиент.Фамилия  
ORDER BY Клиент.Фамилия ASC;
```

 Псевдонимы можно задавать не только для столбцов, но и для таблиц

 Слово **ASC** задает сортировку по возрастанию (его можно опустить).
Для сортировки по убыванию используется слово **DESC**

Оператор SELECT

■ Сравнение с режимом конструктора

Результат запроса теперь такой:

Посмотрим, как созданный запрос выглядит в режиме конструктора

	Фамилия	Количество
▶	Иванов	2
	Николаев	1
	Петров	1
	Федоров	0

The screenshot shows a query builder interface with two tables: 'Клиент' and 'Подписка'. The 'Клиент' table has fields: КодКлиента, Фамилия, Имя, and Отчество. The 'Подписка' table has fields: КодКлиента and КодЖурнала. A relationship line connects the 'КодКлиента' field in 'Клиент' to the 'КодКлиента' field in 'Подписка', with a '1' on the 'Клиент' side and '∞' on the 'Подписка' side. Below the tables is a configuration panel with the following settings:

Поле:	Фамилия	Количество: КодЖурнала
Имя таблицы:	Клиент	Подписка
Групповая операция:	Группировка	Count
Сортировка:	по возрастанию	
Вывод на экран:	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
Условие отбора:		
или:		

Оператор **SELECT**

■ Ограничение результирующих наборов

Оператор **SELECT** извлекает данные из одной или нескольких таблиц и из всех возможных комбинаций оставляет только те, которые соответствуют заданным критериям отбора

Говорят, что при выборке выполняются операции **умножения** и **сужения**

Умножение – построение всех возможных комбинаций записей (**декартово произведение** таблиц)

Сужение – отсеечение «лишних» комбинаций

Умножение выполняется в предложении **FROM** и там же с помощью слова **ON** может быть выполнено сужение (из всех комбинаций записей остаются только связанные между собой)

Оператор SELECT

■ Ограничение результирующих наборов

Для выполнения сужения в операторе **SELECT** могут присутствовать еще два вида предложений:

WHERE и **HAVING**

Предложение **WHERE** <условие> располагается после предложения **FROM** и позволяет наложить дополнительные ограничения на результат объединения.

Ограничения, заданные словом **ON** иногда могут быть перенесены также в предложение **WHERE**

Предложение **HAVING** <условие> может располагаться после предложения **GROUP BY** и применяться к данным в каждой группе

При использовании предложения **HAVING** без предложения **GROUP BY**, оно применяется ко всей результирующей таблице и действует аналогично предложению **WHERE**

Оператор SELECT

■ Ограничение результирующих наборов

Пример. Дополним разработанный ранее запрос новыми ограничениями

```
SELECT Клиент.Фамилия, Count(Подписка.КодЖурнала) AS Количество  
FROM Клиент LEFT JOIN Подписка  
ON Клиент.КодКлиента = Подписка.КодКлиента  
WHERE Left(Клиент.Фамилия,1) = "И" AND Подписка.КодЖурнала > 1  
GROUP BY Клиент.Фамилия  
HAVING Count(Подписка.КодЖурнала) > 0  
ORDER BY Клиент.Фамилия ASC;
```

В результат входят записи только для клиентов, фамилия которых начинается с «И» и при этом используются только журналы, коды которых больше «1».

Кроме того, не включаются клиенты, у которых нет подписки



Внимание! Приведенный пример содержит избыточные условия и составлен исключительно с целью демонстрации использования различных способов ограничения результирующих наборов