6. SQL

Структурированный язык запросов

Появление теории реляционных баз данных и предложенного Коддом языка запросов "alpha", основанного на реляционном исчислении, инициировало разработку ряда языков запросов, которые можно отнести к двум классам:

1. Алгебраические языки, позволяющие выражать запросы средствами специализированных операторов, применяемых к отношениям (JOIN - соединить, INTERSECT - пересечь, SUBTRACT - вычесть и т.д.).
2. Языки исчисления предикатов представляют собой набор правил для записи выражения, определяющего новое отношение из заданной совокупности существующих отношений. Другими словами исчисление предикатов есть метод определения того отношения, которое нам желательно получить (как ответ на запрос) из отношений, уже имеющихся в базе данных.

Разработка, в основном, шла в отделениях фирмы IBM (языки ISBL, SQL, QBE) и университетах США (PIQUE, QUEL). Последний создавался для СУБД INGRES (Interactive Graphics and Retrieval System), которая была разработана в начале 70-х годов в Университете шт. Калифорния и сегодня входит в пятерку лучших профессиональных СУБД. Сегодня из всех этих языков полностью сохранились и развиваются QBE (Query-By-Example - запрос по образцу) и SQL, а из остальных взяты в расширение внутренних языков СУБД только наиболее интересные конструкции.

Рассматриваемый непроцедурный язык SQL (Structured Query Language - структурированный язык запросов) ориентирован на операции с данными, представленными в виде логически взаимосвязанных совокупностей таблиц. Особенность предложений этого языка состоит в том, что они ориентированы в большей степени на конечный результат обработки данных, чем на

процедуру этой обработки. SQL сам определяет, где находятся данные, какие индексы и даже наиболее эффективные последовательности операций следует использовать для их получения: не надо указывать эти детали в запросе к базе данных.

Реализация в SQL концепции операций, ориентированных на табличное представление данных, позволило создать компактный язык с небольшим (менее 30) набором предложений. SQL может использоваться как интерактивный (для выполнения запросов) и как встроенный (для построения прикладных программ). В нем существуют:

- предложения определения данных (определение баз данных, а также определение и уничтожение таблиц и индексов);
- запросы на выбор данных (предложение SELECT);
- предложения модификации данных (добавление, удаление и изменение данных);
- предложения управления данными (предоставление и отмена привилегий на доступ к данным, управление транзакциями и другие). Кроме того, он предоставляет возможность выполнять в этих предложениях:
- арифметические вычисления (включая разнообразные функциональные преобразования), обработку текстовых строк и выполнение операций сравнения значений арифметических выражений и текстов;
- упорядочение строк и (или) столбцов при выводе содержимого таблиц на печать или экран дисплея;
- создание представлений (виртуальных таблиц), позволяющих пользователям иметь свой взгляд на данные без увеличения их объема в базе данных;
- запоминание выводимого по запросу содержимого таблицы, нескольких таблиц или представления в другой таблице (реляционная операция присваивания).

- агрегатирование данных: группирование данных и применение к этим группам таких операций, как среднее, сумма, максимум, минимум, число элементов и т.п.

В SQL используются следующие основные типы данных, форматы которых могут несколько различаться для разных СУБД:

INTEGER

- целое число (обычно до 10 значащих цифр и знак);

SMALLINT

- "короткое целое" (обычно до 5 значащих цифр и знак);

DECIMAL(p,q)

- десятичное число, имеющее p цифр ($0 < p < 16$) и знак; с помощью q задается число цифр справа от десятичной точки ($q < p$, если $q = 0$, оно может быть опущено);

FLOAT

- вещественное число с 15 значащими цифрами и целочисленным порядком, определяемым типом СУБД;

CHAR(n)

- символьная строка фиксированной длины из n символов ($0 < n < 256$);

VARCHAR(n)

- символьная строка переменной длины, не превышающей n символов ($n > 0$ и разное в разных СУБД, но не меньше 4096);

DATE

- дата в формате, определяемом специальной командой (по умолчанию mm/dd/yy); поля даты могут содержать только реальные даты, начинающиеся за несколько тысячелетий до н.э. и ограниченные пятым-десятым тысячелетием н.э.;

TIME

- время в формате, определяемом специальной командой, (по умолчанию hh.mm.ss);

DATETIME

- комбинация даты и времени;

MONEY

- деньги в формате, определяющем символ денежной единицы ($, руб, ...) и его расположение (суффикс или префикс), точность дробной части и условие для показа денежного значения.

Все запросы на получение практически любого количества данных из одной или нескольких таблиц выполняются с помощью единственного предложения SELECT. В общем случае результатом реализации предложения **SELECT** является другая таблица. К этой новой (рабочей) таблице может быть снова применена операция SELECT и т.д., т.е. такие операции могут быть вложены друг в друга. Представляет исторический интерес тот факт, что именно возможность включения одного предложения SELECT внутрь другого послужила мотивировкой использования прилагательного "структурированный" в названии языка SQL. Перед построением запроса необходимо открыть базу данных (таблицы можно не открывать).

6.1 Простейшая команда SELECT:

SELECT * **FROM** DEKANAT!SGRUP

В результате такого запроса будут выведены все поля (на это указывает *) из источника данных, который указывается после инструкции **FROM**, таблицы SGRUP базы данных DEKANAT. Для простоты вместо DEKANAT будем сокращенно писать D. В этом случае имена полей в рабочей таблице будут совпадать с именами полей в исходной таблице.

Если необходимо отобрать данные из части полей таблицы, например, наименование группы и код специальности, то можно указать **список полей**:

SELECT NAME **AS** NAMEGR, NUMSP **FROM** D!SGRUP

При перечислении имен полей после инструкции **AS** можно задать имя поля в рабочей таблице, если оно не указано, то совпадает с именем поля в исходной таблице. Таким образом осуществляется проекция таблицы.

При выборке данных из нескольких таблиц необходимо перечислить список полей и таблиц, входящих в источник данных. Из справочников специальностей (табл.6.1) и групп (табл.6.2) необходимо отобрать наименование группы, наименование и код специальности (табл. 6.3).

Таблица 6.1(SSPEC)

NUMSP	NAME	NUMKAF
1	СП	1
2	КС	1
7	ПС	2

Таблица 6.2(SGRUP)

NUMGR	NAME	NUMSP
1	СП04А	1
3	КС02Б	2
7	ЭКИ04	4

Таблица 6.3 Рабочая таблица

NAMEGR	NUMSP	NAMESP
СП04А	1	СП
КС02Б	2	КС

Имена полей рабочей таблицы не совпадают с именами полей исходных таблиц - они задаются в после фразы **AS**. В исходных таблицах есть **одинаковые имена полей**. В этом случае их необходимо **уточнять именем таблицы**. Для имени таблицы в команде SELECT можно задать **локальный псевдоним**, указав его после имени таблицы в источнике данных через пробел. Локальный псевдоним можно использовать в любом месте запроса. В FoxPro рекомендуется использовать **двухбуквенные** псевдонимы.

SELECT SG.NAME **AS** NAMEGR, SS.NUMSP, SS.NAME **AS** NAMESP
;
FROM D!SGRUP SG, D!SSPEC SS

Если таблицы – источники перечислены через запятую, то рабочая таблица будет иметь вид **декартового произведения** (т.е. всевозможные сочетания

строк исходных таблиц) (табл.6.4). В табл.6.4 слева приведено значение поля NUMSP из таблицы SGRUP для соответствующих строк.

Таблица 6.4

NUMSP	NAMEGR	NUMSP	NAMESP
1	**СП04а**	**1**	**СП**
1	СП04а	2	КС
1	СП04а	7	ПС
2	КС02б	1	СП
2	**КС02б**	**2**	**КС**
2	КС02б	7	ПС
4	ЭКИ04	1	СП
4	ЭКИ04	2	КС
4	ЭКИ04	7	ПС

Отметим, что строки, в которых совпадают значения полей NUMSP, являются актуальными, остальные – лишние, неверные. Лишние строки убираются из рабочей таблицы с помощью **фильтра** (селекция).

Модифицированный запрос будет иметь вид:

SELECT SG.NAME **AS** NAMEGR, SS.NUMSP, SS.NAME **AS** NAMESP;

FROM D!SGRUP SG**,** D!SSPEC SS;

WHERE SG.NUMSP=SS.NUMSP

Условие фильтра задается после инструкции **WHERE.**

Следует помнить, что такой запрос:

- выполняется долго. Сначала формируется декартово произведение, а затем для каждой строки решается вопрос о включении ее в рабочую таблицу в соответствии с выполнением условия фильтра.

- требует много ресурсов. Если объединяются две таблицы по 1000 строк, в декартовом произведении будет содержаться 1 000 000 строк, а при трех таких же таблицах их будет миллиард.

6.2 Использование источника данных с объединенными таблицами

Получить требуемые данные можно быстрее, используя запрос с указанием объединения таблиц при описании источника данных:

SELECT SG.NAME **AS** NAMEGR, SS.NUMSP, SS.NAME **AS** NAMESP;

FROM D!SGRUP SG **INNER JOIN** D!SSPEC SS;

ON SG.NUMSP=SS.NUMSP

Здесь: INNER JOIN – вид объединения таблиц;

ON – условие объединения.

Виды объединения.

INNER JOIN – внутреннее объединение. В рабочую таблицу включаются строки, отвечающие условию объединения (табл.6.5).

Таблица 6.5

NAMEGR	NUMSP	NAMESP
СП04а	1	СП
КС02б	2	КС

LEFT OUTER JOIN – левое внешнее объединение. В рабочую таблицу включаются строки, отвечающие условию объединения и все оставшиеся записи из левой таблицы (т.е. той таблицы, которая стоит слева от LEFT OUTER JOIN) (табл. 6.6). Поля, которые отсутствуют в левой таблице, устанавливаются в неопределенное значение.

Таблица 6.6

NAMEGR	NUMSP	NAMESP
СП04а	1	СП
КС02б	2	КС
ЭКИ04	NULL	NULL

RIGHT OUTER JOIN – правое внешнее объединение. В рабочую таблицу включаются строки, отвечающие условию объединения и все оставшиеся записи из правой таблицы (той таблицы, которая стоит справа от RIGHT OUTER JOIN) (табл. 6.7).

Таблица 6.7

NAMEGR	NUMSP	NAMESP
СП04а	1	СП
КС02б	2	КС
NULL	7	ПС

FULL OUTER JOIN – В рабочую таблицу включаются строки, отвечающие условию объединения и все оставшиеся записи из обеих таблиц.

6.3 Сортировка

Часто требуется, чтобы данные в рабочей таблице были отсортированы. Для этого в конце запроса располагается директива **ORDER BY**.

SELECT NUMSP, NAMEGR, NUMGR **FROM** D!SGRUP **ORDER BY** NUMSP,; NAMEGR **ASC**.

В результате будет сформирована рабочая таблица, содержащая код специальности, наименование группы, код группы. Строки будут отсортированы по возрастанию кода специальности. При одинаковых значениях кода специальности строки будут отсортированы по наименованию групп в алфавитном порядке, поскольку они символьного типа. В качестве параметров сортировки можно также использовать номера полей, по которым будет производиться сортировка, например:

SELECT NUMSP, NAMEGR, NUMGR **FROM** D!SGRUP **ORDER BY** 1,2 **ASC**

Это порядковые номера в списке полей, указанных сразу после SELECT. Кроме того, можно задать порядок сортировки: **ASC** - по возрастанию, **DESC** - по убыванию. Для символьных полей – по алфавиту или в обратном порядке соответственно.

Также можно указать поле по которому будет производится сортировка, если этого поля нет среди полей, указанных в команде SELECT:

SELECT NUMSP, NAMEGR **FROM** D!SGRUP **ORDER BY** NUMSP, **NUMGR**.

6.4 Фильтры

Логическое выражение, расположенное после инструкции **WHERE** называется фильтром и определяет, какие строки из всех отобранных данных будут включены в рабочую таблицу. Если условий (критериев) для фильтрации несколько, они объединяются логическими операциями.

Вычисление критериев фильтра прекращается, если результат становится определенным. Поэтому, на первое место в выражении фильтра необходимо располагать критерии, которые позволят отфильтровать максимальное количество записей, что является очень важным фактором для уменьшения времени выполнения запроса.

В условиях фильтра, кроме традиционных операций отношения, применяются дополнительные условия:

- BETWEEN X1 AND X2 – значение указанного поля должно быть в диапазоне заданных значений;

- IN (X1, X2 …) – в скобках задается список значений, которые может принимать указанное поле;

- IS NULL – значение не определено;

- LIKE X1 - значение указанного поля подобно заданному шаблону (аналогия операции = для строк).

SELECT SG.NAME **AS** NAMEGR, SS.NUMSP, SS.NAME **AS** NAMESP;
FROM D!SGRUP SG **INNER JOIN** D!SSPEC SS;
ON SG.NUMSP=SS.NUMSP;
WHERE NUMSP = 1

В результате выполнения этого запроса в рабочую таблицу отберется только первая строка из табл. 6.5.

Директива **DISTINCT** позволяет подавить размещение в рабочую таблицу совпадающих строк (дубликатов).

SELECT DISTINCT SS.FIO **FROM** D!SSTUD SS

Используется в крайнем случае, так как работает очень медленно. Прежде всего, данные должны быть отсортированы в порядке следования столбцов. В этом случае дубликаты будут располагаться рядом, и для принятия решения о включении строки в рабочую таблицу ее надо сравнить только с последней включенной строкой, а не со всеми.

6.5 Приемники данных

1. **Browse** – приёмник по умолчанию, рабочая таблица создаётся в памяти и загружается в окно **«browse»**.

2. **Cursor** (**Current set of records**) - текущий набор записей. Рабочая таблица записывается в файл. Файл открывается в рабочей области. Рабочая область становится активной. При закрытии cursor'а или рабочей области файл с диска удаляется. Курсор должен иметь псевдоним - **Alias** рабочей области, который используется в программе. С ним можно работать как с обычной таблицей, но надо помнить, что его статус - **ReadOnly**.

SELECT * **FROM** D!SGRUP **INTO CURSOR** SG_C1

3. **Table** – аналогично курсору, но таблица **сохраняется** на диске.

SELECT * **FROM** D!SGRUP **INTO TABLE** SG_T1,

Где файл с именем **SG_T1** будет иметь расширение **DBF**.

4. **Array** – таблица записывается в двумерный массив. Если не была отобрана ни одна строка – массив не создаётся.

SELECT * **FROM** D!SGRUP **INTO ARRAY** SG_AR1

5. **Screen** – Результат запроса выводится на экран. Отобранные данные печатаются на экране в главном окне Visual FoxPro.

SELECT * **FROM** D!SGRUP **TO SCREEN**

6. **Printer** – Вывод на принтер – альтернативный приемник данных для экрана.

SELECT * **FROM** D!SGRUP **TO PRINTER**

7. **File** - Вывод в файл. Второй альтернативный приемник данных.

SELECT * **FROM** D!SGRUP **TO FILE** SG_F1 [**ASCII**]

Если задать в конце запроса ASCII, то рабочая таблица запишется в файл в DOS-кодировке. Вывод данных в файл используется для связи с другими СУБД.

При выводе данных на экран или один из альтернативных приемников в первой строке выводятся наименования полей, а затем строки с данными. Если в конце любого из описанных запросов написать **PLAIN**, то заголовки столбцов выводиться не будут.

SELECT * **FROM** D!SGRUP **TO FILE** SG_F1 **PLAIN**

При передаче данных в альтернативные приемники, параллельно производится их вывод на экран. Директива **NO CONSOLE** – подавляет вывод на экран.

8.**Report** – Позволяет передать отобранные данные в программу формирования отчета по файлу шаблона (**.FRX**). Для этого используется следующие две команды:

SELECT * **FROM** D!SGRUP **INTO CURSOR** SG_C1

REPORT FORM FileName1 **PREVIEW**

Где, FileName1 - задает имя файла описания отчета, по которому печатается отчет;

PREVIEW – директива определяет передачу сформированного отчета не на принтер, а в окно предварительного просмотра.

- **Graph** - передача результата выполнения запроса в **Microsoft Graph**, отдельное приложение, включенное в состав Visual FoxPro
- **Label** - передать данные в программу печати этикеток по шаблону (.**LBX**).

6.6 Конструктор запросов

Запросы создаются для различных целей: отбор информации для отчета, для быстрого поиска ответа на вопрос или для просмотра некоего подмножества данных. Вне зависимости от цели запроса, основные операции при его создании остаются теми же. После выяснения того, какая информация должна быть найдена и какое представление или таблица (или таблицы) хранят ее, создание запроса можно условно разбить на следующие шаги.

1. Начать создание запроса с помощью мастера запроса или конструктора запросов.
2. Определить таблицы, входящие в источник данных и установить связи между ними.

2. Выбрать, какие поля должны содержаться в рабочей таблице.

3. Установить критерий поиска записей, содержащих требуемые результаты.

4. Определить опции сортировки или группирования данных в запросе.

5. Выбрать тип приемника данных запроса: таблица, представление, отчет, просмотр и т.д.

6. Выполнить запрос.

В FoxPro имеется конструктор запросов. С его помощью можно быстро и эффективно составлять запросы, просматривать сгенерированный код, формировать источники данных и т.д.

Запустить конструктор запросов можно как из диспетчера проектов, так и из меню File.

1. Выделите вкладку **Data** в диспетчере проектов.
2. Выделите **Queries**.
3. Выберите **New Query**.

Конструктор запросов также можно запустить, выбрав **New** в меню **File**, выделив опцию **Query** и выбрав **New File.** При создании нового запроса вам будет предложено указать таблицу или представление из текущей базы данных или свободной таблицы.

В конструктор, вид которого представлен на рис.6.1, необходимо добавить таблицы-источники. Из этих таблиц выбираются данные (рис.6.2).

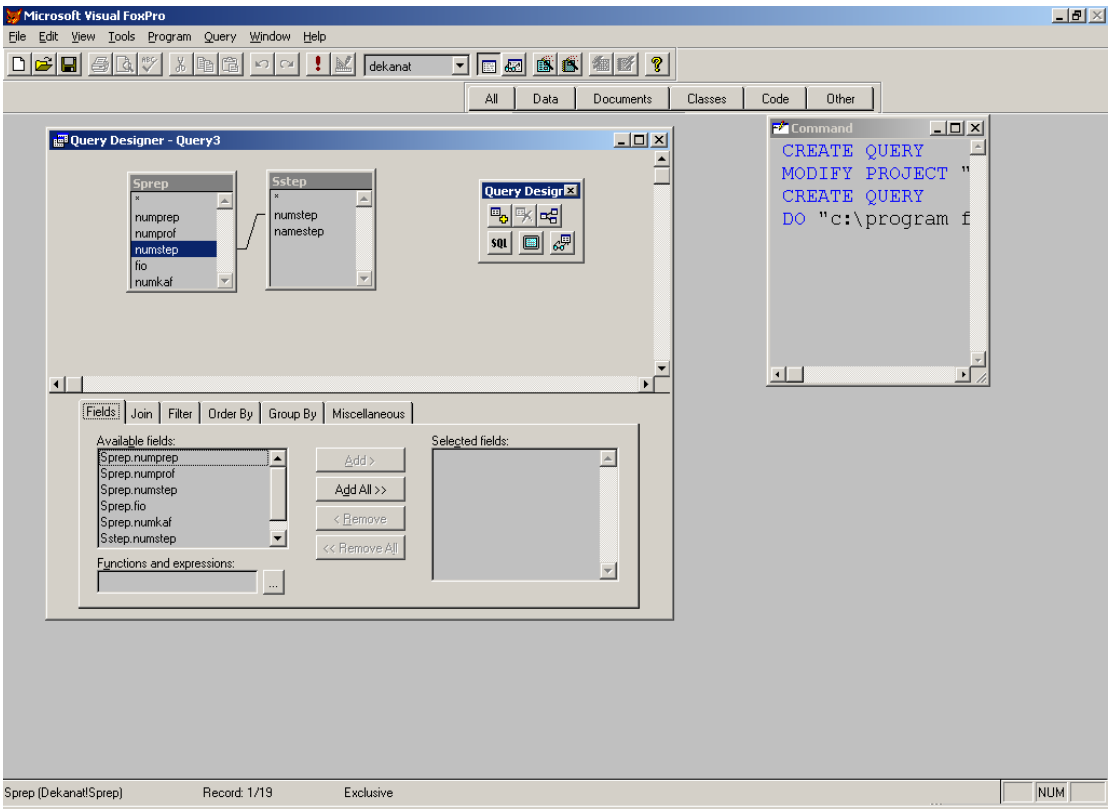

Рис.6.1 Вид окна конструктора запросов.

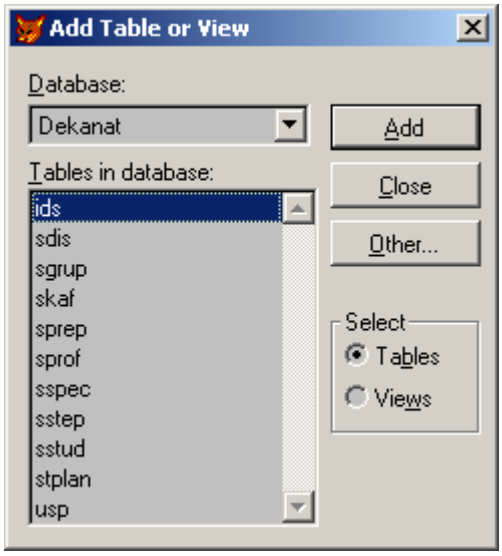

Рис.2.2 Окно выбора таблиц – источников

Назначение вкладок конструктора запросов и конструктора представлений

JOIN - Selection Criteria

Задаются условия выбора записей при объединении таблиц, условия объединения, определяющие временные отношения между таблицами.

Fields

Задаются поля, агрегирующие функции (такие, как SUM или COUNT) или другие выражения.

Order By

Задаются поля, агрегирующие функции или другие выражения, которые используются для упорядочения записей, извлекаемых в ходе запроса.

Group By

Задаются поля, агрегирующие функции или другие выражения, которые используются для комбинирования записей в группу исходя из идентичных значений в этих полях.

Miscellaneous

Задаются параметры запроса – подавление дубликатов, сохранение в рабочей таблице части строк, условия обновления представлений, (**Update Criteria** - только в конструкторе представлений) и т.п.

Замечание При добавлении таблиц в конструкторе FoxPro автоматически установит связи между ними. Однако, они не всегда корректные. Поэтому их надо разрывать и устанавливать связи последовательно от родительской таблицы к дочерней. Возможности конструктора по объединению таблиц в Visual FoxPro 6.0 ограничены. В нем можно реализовать только линейную схему данных.

В конструктор не удастся организовать сложные связи между таблицами (рис.6.3).

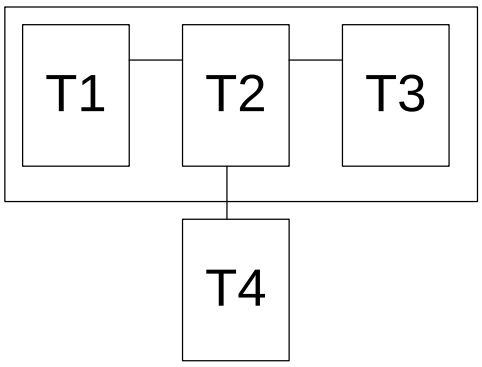

Рис. 6.3

1-й способ описания такого источника данных.

Для организации такой связи меду таблицами можно использовать схему:

T4, T1 **INNER JOIN** T2 **INNER JOIN** T3 **ON** T2=T3 **ON** T1=T2;

WHERE T2 = T4

Стоящее после запятой объединение представляет собой источник данных из трех таблиц, которые образуют декартово произведение с четвертой таблицей. Однако, такой запрос не будет оптимальным, поскольку требуется много ресурсов и времени на его выполнение.

2-й способ.

До сих пор мы говорили о таблицах, которые *реально* хранятся в базе данных. Это, так называемые, базовые таблицы. Существует другой вид таблиц, получивший название "представления" (иногда их называют "представляемые таблицы" или виртуальные) (рис.6.4).

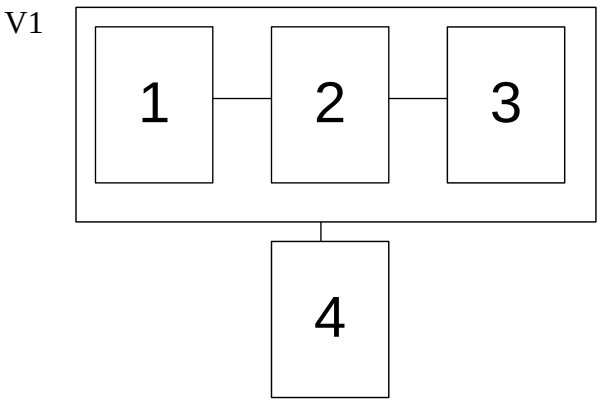

Рис. 6.4

Представление (view) - это таблица, содержимое которой берется из других таблиц посредством запроса. При этом новые копии данных не создаются. Описания представления содержаться в файле DBC. При обработке представления таблицы открываются, устанавливаются все связи между ними, настраиваются фильтры. Схема описания представления V1:

T1 **INNER JOIN** T2 **INNER JOIN** T3 **ON** T2=T3 **ON** T1=T2

Представление V1 можно добавить в источник данных запроса и объединять с базовыми таблицами:

V1 **INNER JOIN** T4 **ON** T4=V1

3-й Способ:

В описании источника данных используются скобки (наиболее предпочтительный способ):

(T1 **INNER JOIN** T2 **INNER JOIN** T3 **ON** T2=T3 **ON** T1=T2);

INNER JOIN T4 **ON** T4=T2

6.7 Запросы с группировкой данных

Пусть имеется таблица учебного плана STPLAN(табл.6.7). И имеется таблица успеваемости студентов USP (табл.6.8)

Таблица 6.7 (STPLAN)

NUMPL	NUMGR	NUMPREP	NUMDIS	SEMESTR
1	1	7	17	5
2	2	7	17	5
3	1	9	19	3

Таблица 6.8(USP)

NUMPL	NUMST	REZ
* 1	1	5
2	1	3
* 1	2	3
* 1	3	2

Необходимо составить запрос, в котором формируются итоговые результаты экзаменов по группам, дисциплинам и преподавателям. В этом запросе необходимо по всем записям в таблице USP, имеющим одинаковые значения в поле NUMPL, просуммировать количества «5», «4», «3», «2» и рассчитать средний балл по группе. В табл. 6.8 такие записи помечены *. В результате формируется курсор, содержащий наименование группы, ФИО преподавателя, наименование дисциплины, которую он читает, средний балл по группе, и количества «5», «4», «3», «2». В нем для N строк из таблицы USP с одинаковыми значениями в поле NUMPL будет получена одна строка с подсчитанными итогами. Вид этого курсора представлен в табл. 6.9.

Таблица 6.9

NAMEGR	FIO	NAMEDIS	BALSR	R5	R4	R3	R2
СП04а	Лапко В.В.	ТПЦВМ	4,5	2	2	0	0

СП06а	Ковалев	ПТЦА	3,5	1	1	1	1

Источник данных для запроса представлен на рис.6.5.

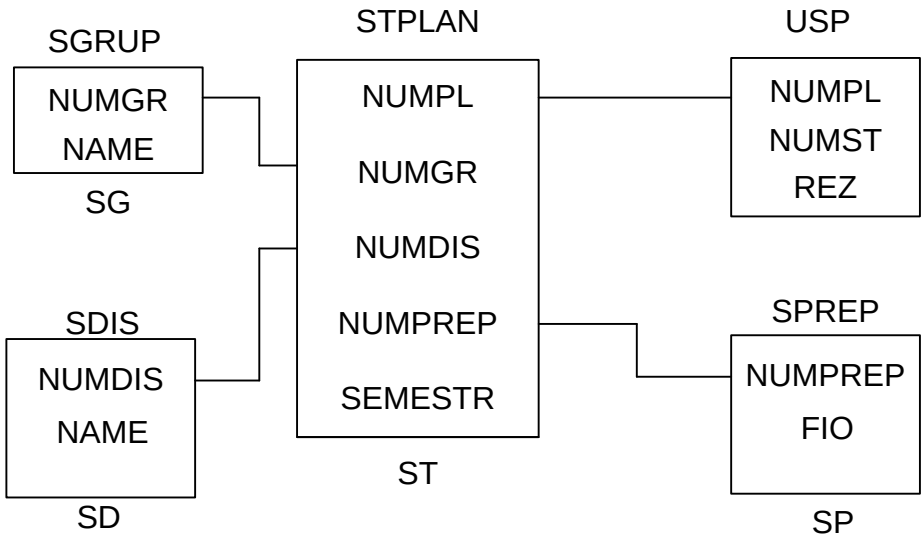

Рис. 6.5

Команда SELECT будет иметь вид:

SELECT SG.NAME **AS** NAMEGR, FIO, SD.NAME **AS** NAMEDIS, **AVG**(REZ) **AS** ;

BALSR, **SUM**(**IIF**(REZ=5),1,0) **AS** R5, **SUM**(**IIF**(REZ=4),1,0) **AS** R4, ; **SUM**(IIF(REZ=3),1,0) **AS** R3, **SUM**(**IIF**(REZ=2),1,0) **AS** R2 ;

FROM ((D!USP **INNER JOIN** D!STPLAN ST **INNER JOIN** SGRUP SG ;

ON ST.NUMGR=SG.NUMGR;

ON USP.NUMPL=ST.NUMPL);

INNER JOIN D!SDIS SD **ON** SD.NUMDIS=ST.NUMDIS) **INNER JOIN** ;

D!SPREP SP ;

ON SP.NUMPREP=ST.NUMPREP;

GROUP BY USP.NUMPL

Выполнение запроса с группировкой данных производится следующим образом:

- Исходные данные сортируются в порядке перечисления полей в команде **GROUP BY** для того, чтобы все строки с одинаковыми значениями в перечисленных полях располагались рядом.
- Осуществляется проход по всем строкам с совпадающими значениями в указанных полях, производятся заданные вычисления.
- При изменении значения в одном из перечисленных полей формируется одна строка в выходном курсоре, в которой размещаются неизменяемые значения полей для группировки и рассчитанные итоговые значения.
- Указанные операции повторяются для всех сгруппированных значений.

В результате выполнения такого запроса получается курсор со следующими свойствами:

- При использовании группировки данные будут отсортированы по полю (полям), указанному в **GROUP BY.**
- В выходном курсоре не будет дубликатов.

Если полученная сортировкам данных не устраивает пользователя, можно отсортировать выходной курсор, добавив в конце запроса инструкцию сортировки, например, **ORDER BY NAMEGR.**

К сгруппированным строкам можно применить фильтр **HAVING.** Действие его аналогично фильтру **WHERE.** Основное отличие заключается в том, что он применяется для сгруппированных строк, т.е. в критерии фильтра можно указать условия, включающие значения вычисляемых полей, например:

… **GROUP BY** USP.NUMPL **HAVING** FIO LIKE ‘Ковалев’

или

… **GROUP BY** USP.NUMPL **HAVING** FIO LIKE ‘Ковалев’ **AND** BALSR >= 4

В первом примере будет произведен отбор групп, экзамены в которых проводит преподаватель Ковалев. Эту задачу можно выполнить и с помощью простого фильтра **WHERE**. Во втором примере добавлен критерий включения строк для групп, имеющих средний балл 4 и выше. Такое условие можно проверить только при использовании инструкции **HAVING.**

Агрегатные функции

Это функции, применяемые для подсчета каких-либо значений, заранее запрограммированные, производящие числовые операции. В Visual FoxPro к таким функциям относятся SUM(), AVG(), MIN(), MAX(), COUNT().

COUNT() – счетчик, подсчитывает количество строк в группе.

AVG() - среднее арифметическое.

SUM() - определяет сумму значений из указанного столбца.

MAX() - возвращает максимальное значение.

MIN() - возвращает минимальное значение.

В запросе с группировкой данных все поля не указанные в инструкции **GROUP BY** должны входить в агрегатную функцию. В Visual FoxPro 6.0 это требование не является обязательным. Если разработчику известно, что значение такого поля в группе не будет изменяться, применять агрегатную функцию не обязательно. В приведенном запросе это поля SG.NAME, FIO и SD.NAME. В других диалектах, включая Visual FoxPro 8 и старше, применение агрегатной функции обязательно.

6.8 Вложенные запросы, подзапросы

Для решения задач, требующих сложной обработки данных, необходимо использовать такое средство как вложенные запросы, называемые также подзапросами.

Подзапрос представляет собой такой же SQL-запрос, начинающийся со слова SELECT, который формирует список каких-либо значений из других таблиц. Подзапрос всегда заключается в круглые скобки. В основном запросе в критерии фильтра производится проверка вхождения значения указанного поля в список значений, сформированный подзапросом.

Требуется отпечатать список групп кафедры ЭВМ, которые сдавали экзамены хотя бы раз. Информация о результатах экзаменов заносится в таблицу успеваемости (**USP**). Следовательно, если студенты группы не сдавали экзамены, информации о таких группах в таблице успеваемости не будет, они не должны включаться в выходной курсор. Ниже приведен пример реализации запроса без использования подзапроса:

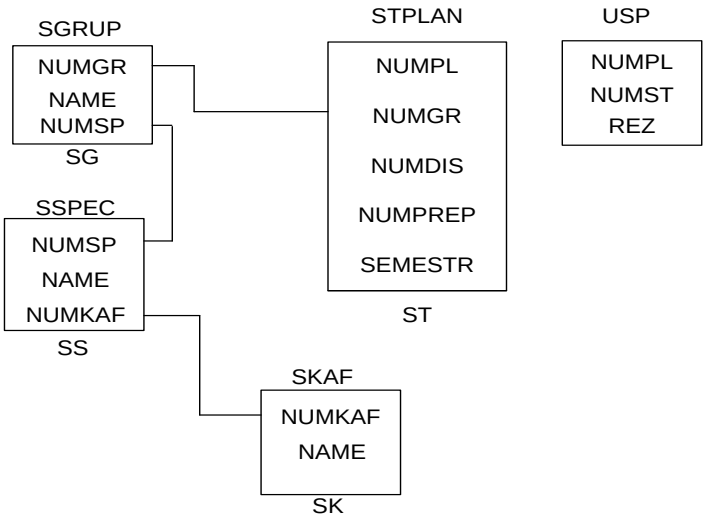

рис.6.6 Источник данных

Команда запроса

```
SELECT DISTINCT SG.NAME;
FROM D!USP, D!STPLAN ST INNER JOIN  D!SGRUP SG ;
      INNER JOIN D!SSPEC SS  INNER JOIN D!SKAF SK;
      ON SS.NUMKAF=SK.NUMKAF;
      ON SG.NUMSP=SS.NUMSP;
      ON ST.NUMGR=SG.NUMGR;
WHERE USP.NUMPL=ST.NUMPL AND SK.NAME LIKE ‘ЭВМ’
```

Поскольку источники перечислены через запятую, то в результате получается декартово произведение, для того чтобы отфильтровать все лишние строки, используется критерий фильтра **USP.NUMPL=ST.NUMPL**. Чтобы наименование группы в курсоре встречалось один раз, используется **DISTINCT**.

Можно также объединить (**INNER JOIN**) таблицы **STPLAN** и **USP**, чтобы не использовать декартово произведение. Если в строках не выполняется условие объединения, т.е. в таблице **STPLAN** встречаются значения в поле **NUMPL,** которых нет в соответствующем поле таблицы **USP**, то данные из текущей строки STPLAN не будут включены в выходной курсор.

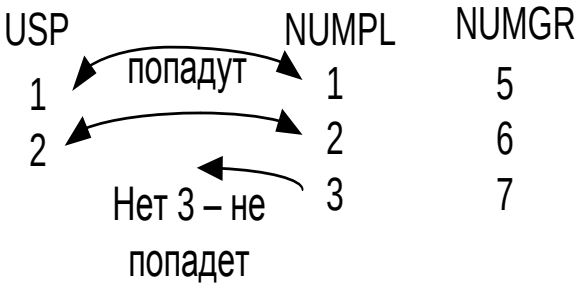

Рис. 6.7

Приведенная команда неудобная, ненаглядная, в ней нет гибкости, о чем будет сказано ниже. Кроме того, далеко не все запросы можно написать с использованием только фильтра.

Пример выполнения этого же запроса, но **с использованием подзапроса**.

Источник данных для запроса приведен на рис.6.8.

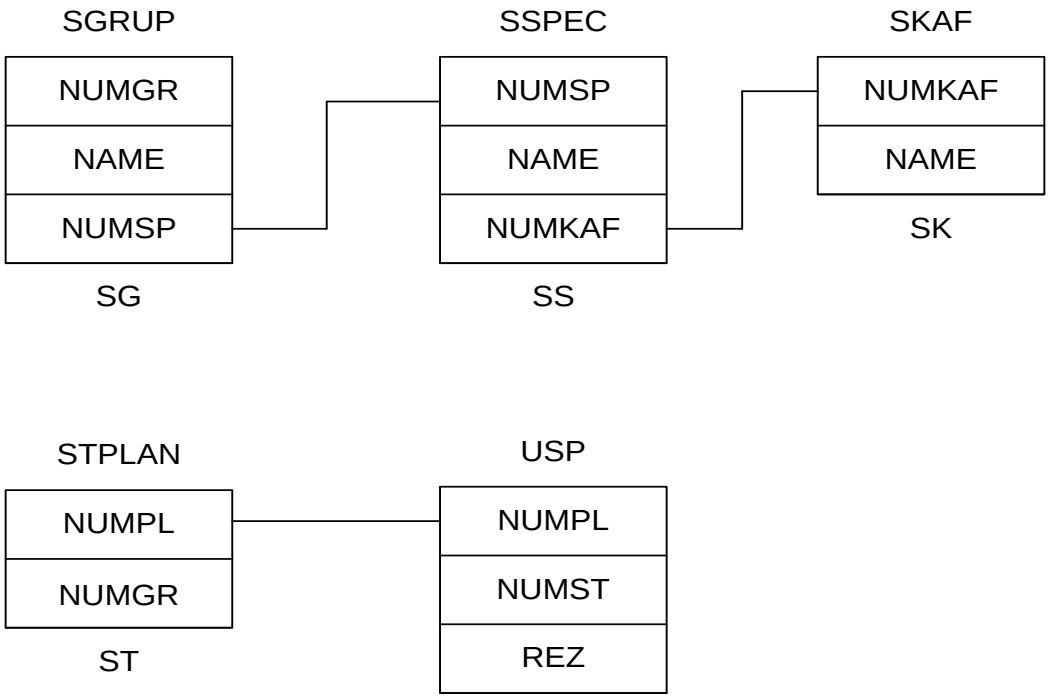

Рис.6.8

SELECT SG.NAME **AS** NAMEGR;

FROM D!SGRUP SG **INNER JOIN** D!SSPEC SS **INNER JOIN** D!SKAF SK;

ON SS.NUMKAF=SK.NUMKAF;

ON SG.NUMSP=SS.NUMSP;

WHERE SG.NUMGR [NOT] **IN** ;

(**SELECT DISTINCT** ST.NUMGR;

FROM D!STPLAN ST **INNER JOIN** D!USP;

ON USP.NUMPL=ST.NUMPL) **AND**

SK.NAME=‘ЭВМ’

В начале выполняется подзапрос, отобранные коды групп, студенты которых сдавали экзамены, сохраняются в памяти. При работе внешнего запроса перед включением строки в курсор проверяется вхождение значения заданного поля SG.NUMGR в созданный список.

Запрос с подзапросом работает дольше, чем запрос с фильтром. Однако, они очень удобны, например, если надо отобрать группы первокурсников, не сдававшие сессию ни разу, то достаточно поставить NOT в условии фильтра. Другим способом (с использованием фильтра) это сделать гораздо тяжелее, а иногда и невозможно.

6.9 Коррелированный запрос

.

Если в подзапросе содержится ссылка на значение, получаемое из возможного кортежа блока запроса более верхнего уровня, то запрос называется **коррелированным подзапросом**. Так как их результат зависит от значений, определенных во внешнем подзапросе, то обработка коррелированного подзапроса, следовательно, должна повторяться для каждого значения извлекаемого из внешнего запроса, а не выполняться раз и навсегда.

Пример коррелированного запроса для формирования списка групп студентов, сдававших экзамены:

SELECT SG.NAME **AS** NAMEGR;

FROM D!SGRUP SG **INNER JOIN** D!SSPEC SS **INNER JOIN** D!SKAF SK;

ON SS.NUMKAF=SK.NUMKAF;

ON SG.NUMSP=SS.NUMSP;

WHERE EXIST;

(**SELECT** ST.NUMGR;

FROM D!STPLAN ST, D!USP;

WHERE USP.NUMPL=ST.NUMPL AND;

SG.NUMGR=ST.NUMGR) **AND** SK.NAME = 'ЭВМ'

Коррелированный подзапрос выполняется для каждой строки, когда решается вопрос о включении ее в выходной курсор. В коррелированном подзапросе нельзя применять объединение таблиц. Поэтому, такой запрос всегда выполняется очень долго.

6.10 Объединение запросов

Предположим, надо отпечатать список приглашенных на день кафедры. Среди приглашенных будут преподаватели и студенты. Список составляется по форме, приведенной в табл.6.10. Для преподавателей необходимо привести фамилию и должность, а для студентов фамилию и группу. И все они должны быть, непременно, с кафедры ЭВМ. У студентов нет должности, в выходной курсор включается константа «Студент». У Преподавателей нет значений для поля «Группа», поле остается пустым.

Таблица 6.10

FIO	PROF	GROUP
ЛАПКО В.В.	ДОЦЕНТ	
ГУСЕВ Б.С.	ДОЦЕНТ	
...	....	...
ПЕТРОВ В.В.	СТУДЕНТ	КС04А
ЕГЕРЬ	СТУДЕНТ	КС06В

Данные по преподавателям и студентам находятся в разных таблицах. Источники данных для выполнения запроса приведены на рис.6.9. Для отбора данных из разных источников составляются две команды **SELECT**. Объединение данных в один курсор выполняется с помощью инструкции **UNION**. При этом данные из разных таблиц, помещаемые в выходной курсор, должны быть однотипные. Строки должны иметь поля одинаковые по типу и длине. Так как длина поля FIO в справочнике студентов на 5 символов больше, чем в справочнике преподавателей, к фамилии преподавателей добавляется 5 пробелов. Поле «Группа» в источнике данных для преподавателей отсутствует, поэтому, вместо него в запрос вставляется «заглушка», состоящая из 10 пробелов.

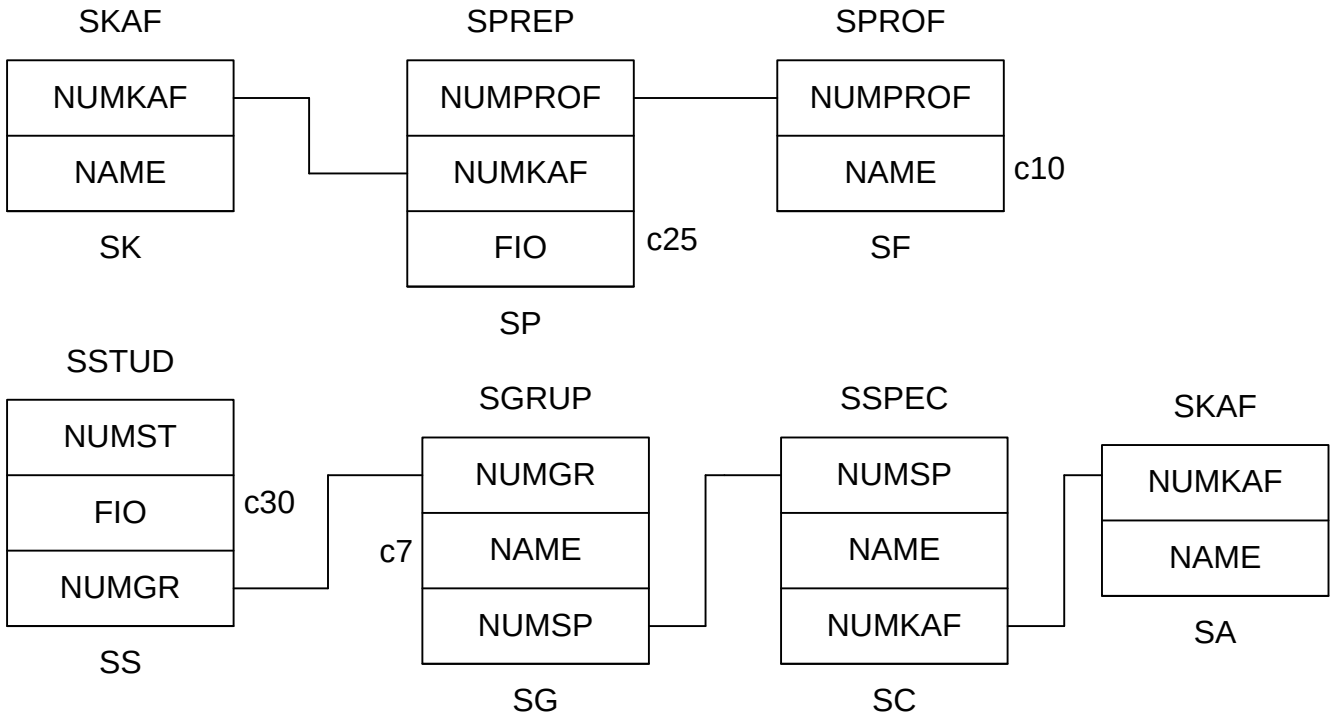

Рис.6.10 Источник данных

Пример объединения запросов.

SELECT SP.FIO+’ ’ **AS** FIO, SF.NAME **AS** PROF, ‘ ’ **AS** GROUP;

FROM D!SKAF SK **INNER JOIN** D!SPREP SP **INNER JOIN** D!SPROF SF;

ON SP.NUMPROF=SF.NUMPROF;

ON SK.NUMKAF=SP.NUMKAF;

WHERE SK.NAME=’ЭВМ’;

UNION [ALL];

SELECT SS.FIO, ‘СТУДЕНТ’, SG.NAME;

FROM D!SSTUD SS **INNER JOIN** D!SGRUP SG **INNER JOIN** D!SSPEC SC ; **INNER JOIN** D!SKAF SA ;

ON SC.NUMKAF=SA.NUMKAF;

ON SG.NUMSP=SC.NUMSP;

ON SS.NUMGR=SG.NUMGR;

WHERE SA.NAME=’ЭВМ’;

ORDER BY 1,2

Таким образом, будет отобран список приглашенных. Для сортировки в конец команды можно добавить ORDER BY 1,2.

Основные правила построения запросов с объединением

Все команды SELECT создают свои промежуточные результирующие наборы, которые затем объединяются в один курсор.

1. Можно объединять до **10 членов UNION**.
2. В первой команде SELECT **определяется структура полей** выходного курсора (количество, порядок, тип и размер).
3. Имена соответствующих столбцов в разных командах SELECT могут быть разные. Несоответствие типов полей приведет к ошибке. В **первой команде SELECT определяются имена полей** в курсоре.
4. Если **поля** в какой-либо инструкции SELECT **нет**, то ставится заглушка.
5. **Заглушка в первой SELECT** резервирует поля для остальных инструкций. Она должна указывать на тип поля и его размер. Заглушка для цифрового поля: 0000 – если целое, 00.00 – если дробное, .F. – логического типа, { ^ / / } – типа дата.
6. Если в первой инструкции SELECT **длина поля будет меньше**, чем в остальных, то будет осуществляться усечении длины данных в других инструкциях.

7. Инструкции **GROUP BY, WHERE, HAVING** можно использовать в каждой SELECT. Они оказывают влияние только на записи, относящиеся к данной команде.
8. Инструкции **ORDER BY** и **INTO** нужно поместить в последнюю инструкцию SELECT. В инструкции ORDER BY указываются не имена полей, а их порядковые номера (нумерация начинается с 1). Эти инструкции допускается размещать в любую SELECT, но только один раз, они будут распространены на весь курсор. Если они встретятся два или более раз – это приведет к ошибке.
9. Если не указан порядок сортировки, то данные будут отсортированы в порядке следования полей.
10. UNION аналогично **DISTINCT,** что замедляет выполнение запроса. Если есть уверенность, что дубликатов нет или их появление не существенно, инструкцией **ALL** можно отменить проверку на дублирование строк.
11. С помощью членов UNION нельзя объединять результаты подзапросов.

6.11 Запрос с самообъединением таблиц.

Пусть необходимо сформировать список сотрудников и их начальников, при условии, что все данные хранятся в одной таблице (табл.6.11). База данных имеет имя B.

Таблица 6.11

IDS	FIO	IDB
1	Петров	1
2	Сидоров	1
3	Егоров	2
4	Анаев	2

Первое поле IDS – это код сотрудника. FIO – Фамилия, имя, отчество сотрудника. IDB – код сотрудника, являющегося начальником данного. Таким образом: начальником Анаева является сотрудник с кодом 2 – Сидоров, начальником Сидорова является Петров, т.к. IDB=1 – Петров. Из этой таблицы формируется список начальников и подчиненных (табл.6.12).

Таблица 6.12

IDS	FIOS	FIOB
1	Петров	Петров
2	Сидоров	Петров
3	Егоров	Сидоров
4	Анаев	Сидоров

Источник данных для этого запроса представлен на рис.6.11. Очевидно, что в этом случае необходимо открывать одну и ту же таблицу два раза. В запросе допускается открывать таблицу более одного раза, необходимо только назначить всем копиям разные псевдонимы.

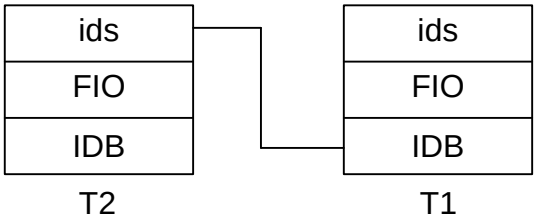

Рис. 6.11

Команда SELECT:

SELECT T1.IDS, FIO **AS** FIOS, T2.FIO **AS** FIOB;
FROM B!T T1 **INNER JOIN** B!T T2 **ON** T1.IDB=T2.IDS

6.12 Параметризированные критерии запроса.

Как правило, запросы являются универсальными. Это означает, что с помощью одного запроса можно отбирать данные не для одной кафедры ЭВМ, а для любой кафедры, сведения о которой имеются в БД.

В общем случае фильтр для выбора кафедры будет иметь вид:

WHERE NAMEKAF **LIKE** cNameKaf

Чтобы использовать переменную в фильтре, необходимо, чтобы она была объявлена, видна, и ей должно быть присвоено значение.

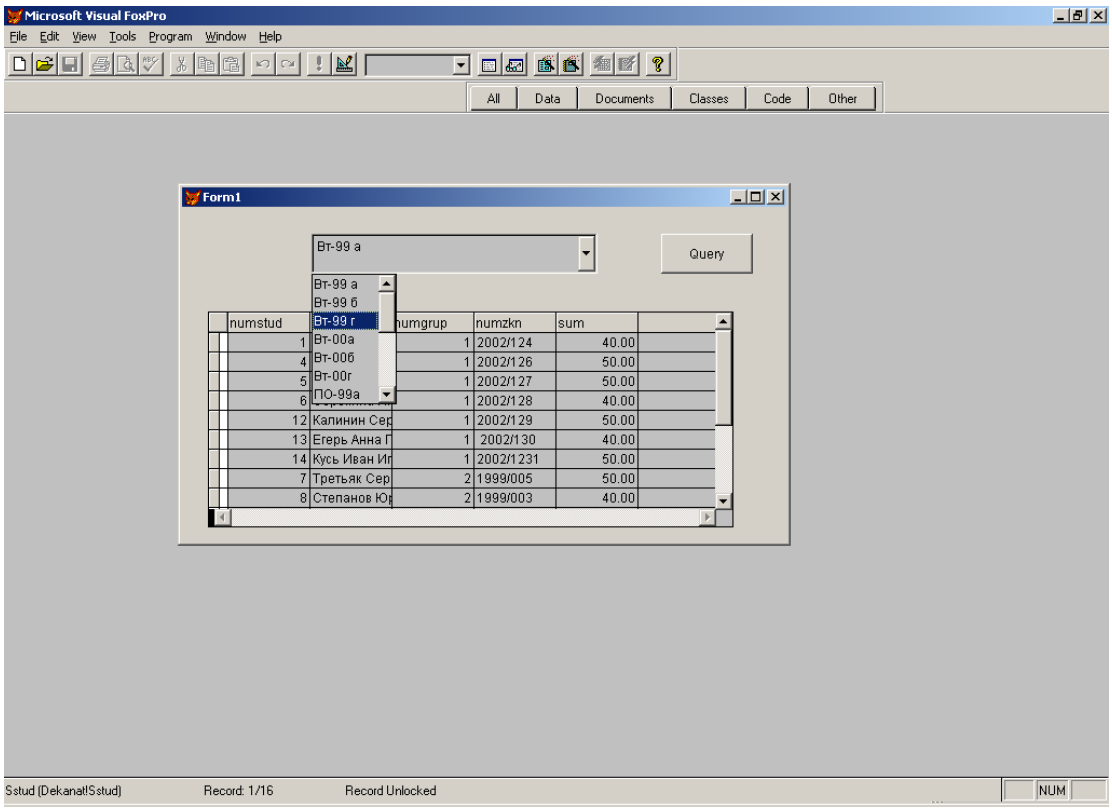

Рис. 6.12

На рис. 6.12 представлен вид формы, с помощью которой можно получить данные по выбранной группе. Для этого в методе OnClick кнопки (Button) QUERY объявляется переменная, и она используется для запроса. Если перед именем переменной поставить «?» – то это будет параметр. Например,

WHERE NAMEKAF **LIKE** ?cNameKaf

Перед выполнением запроса FoxPro «попросит» ввести значение этого параметра (это используется только в представлениях). Однако, это хорошо для отладки программы, и не стоит в реальной БД использовать этот метод – это вызовет массу неудобств для пользователя.

6.14 Перекрестные запросы

В таблице успеваемости USP данные о результатах сессии хранятся в виде, представленном в табл. 6.14.

Таблица 6.14

NUMST	NUMPL	REZ
1	1	5

Используя запрос, можно сформировать курсор, в котором разместятся фамилии студентов и дисциплины (табл.6.15).

Таблица 6.15

FIO	DIS	REZ
Петров	ТОЭ	5
Петров	ОБД	4
Петров	ТП	4
Егоров	ТП	3
Егоров	ТОЭ	4
Егоров	ОБД	3

Однако такая таблица не отличается наглядностью. Гораздо удобнее будет таблица вида табл.6.16.

Таблица 6.16

	ТОЭ	ОБД	ТП
Петров	5	4	4
Егоров	4	3	3

Запросы, формирующие таблицы такого рода называются **перекрестными.**

Чтобы сформировать перекрестную таблицу необходимо сформировать промежуточный курсор (табл.6.15), в котором будут содержаться наименования строк и столбцов перекрестной таблицы и столбец результатов. Данные в этом курсоре необходимо отсортировать по наименованию строк и столбцов.

Источник данных для такого запроса представлен на рис. 6.13.

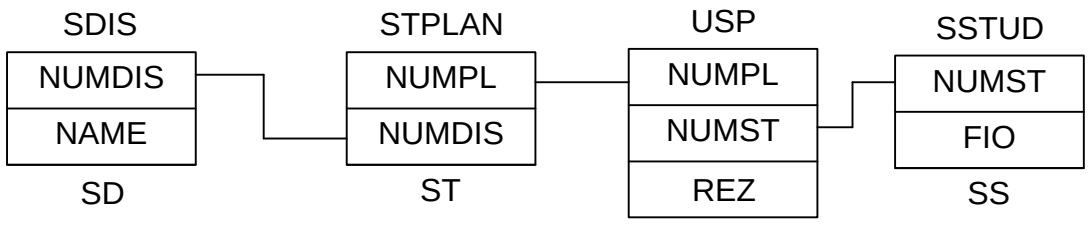

рис. 6.13

SELECT SS.FIO, SD.NAME **AS** DIS, USP.REZ;

FROM D!SDIS SD **INNER JOIN** D!STPLAN ST **INNER JOIN** D!USP;

INNER JOIN D!SSTUD SS;

ON USP.NUMST = SS.NUMST;

ON ST.NUMPL = USP.NUMPL;

ON SD.NUMDIS=ST.NUMDIS;

ORDER BY FIO, DIS ;

INTO CURSOR MY_C

DO (_GENXTAB)

После формирования промежуточного курсора выполняется запуск программы, формирующей перекрестную таблицу. **_GENXTAB** – системная переменная, в которой хранится имя программы, формирующей перекрестную таблицу. В эту программу можно передавать параметры:

DO (_GENXTAB) **WITH** ‘C1_1’,….

В параметрах можно передать:

1. Имя курсора.
2. Создавать ли только курсор (по умолчанию .F.). Создает курсор или таблицу.
3. Закрывать ли входную таблицу после использования (по умолчанию .T.).
4. Отображать ли «градусник» (индикатор выполнения процесса) (по умолчанию .T.).
5. Номер поля наименования строки (по умолчанию 1).
6. Номер поля наименования столбца (по умолчанию 2).
7. Номер поля данных (по умолчанию 3).
8. Вычислять ли итоги по строкам (.F. – по умолчанию).
9. Варианты вычислений:
   - 0 – сумма по строке
   - 1 – количество заполненных столбцов
   - 2 – процент от общей суммы.

После формирования перекрестной таблицы командой **BROWSE NO MODIFY** можно загрузить полученный курсор в окно просмотра.