Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Заместитель директора
по учебной работе
_____Малафей И.В.
«___»_____2016

Специальность: «Программное обеспечение информационных технологий»	Дисциплина: «Базы данных и системы управления базами данных»
Составлена на основании учебной программы, утвержденной директором Колледжа бизнеса и права 25.11.2011	

Лабораторная работа № 7

Инструкционно-технологическая карта

Тема: Создание простых запросов на выборку
Цель работы:

–научиться создавать простые запросы на выборку с использованием инструкции SELECT: отбор строк (предложение WHERE), выборка всех столбцов (инструкция SELECT *), выборка одной строки, удаление повторяющихся строк (предикат DISTINCT), сортировка результатов запроса (предложение ORDER BY);

–сформировать умения создания простых запрос с использованием всех видов условий отбора;

–изучить понятия «подстановочные знаки» и «вычисляемый столбец»;

–изучить на примерах особенности реализации простых запросов на выборку.

Время выполнения: 2 часа

Краткие теоретические сведения

Инструкция SELECT

Инструкция SELECT извлекает информацию из базы данных и возвращает ее в виде таблицы результатов запроса.

На рисунке 1 приведена синтаксическая диаграмма инструкции SELECT. Инструкция состоит из шести предложений. Предложения SELECT и FROM являются обязательными. Четыре остальных включаются в инструкцию только при необходимости. Ниже перечислены функции каждого из предложений.

- В предложении **SELECT** указывается список столбцов, которые должны быть возвращены инструкцией SELECT. Возвращаемые столбцы могут содержать значения, извлекаемые из столбцов таблиц базы данных, или значения, вычисляемые во время выполнения запроса.

- В предложении **FROM** указывается список таблиц, которые содержат элементы данных, извлекаемые запросом.
- Предложение **WHERE** показывает, что в результаты запроса следует включать только некоторые строки. Для отбора строк, включаемых в результаты запроса, используется *условие отбора.*
- Предложение **GROUP BY** позволяет создать итоговый запрос. Обычный запрос включает в результаты запроса по одной записи для каждой строки из таблицы. Итоговый запрос, напротив, вначале группирует строки базы данных по определенному признаку, а затем включает в результаты запроса одну итоговую строку для каждой группы.
- Предложение **HAVING** показывает, что в результаты запроса следует включать только некоторые из групп, созданных с помощью предложения GROUP BY. В этом предложении, как и в предложении WHERE, для отбора включаемых групп используется условие отбора
- Предложение **ORDER BY** сортирует результаты запроса на основании данных, содержащихся в одном или нескольких столбцах. Если это предложение не указано, результаты запроса не будут отсортированы.

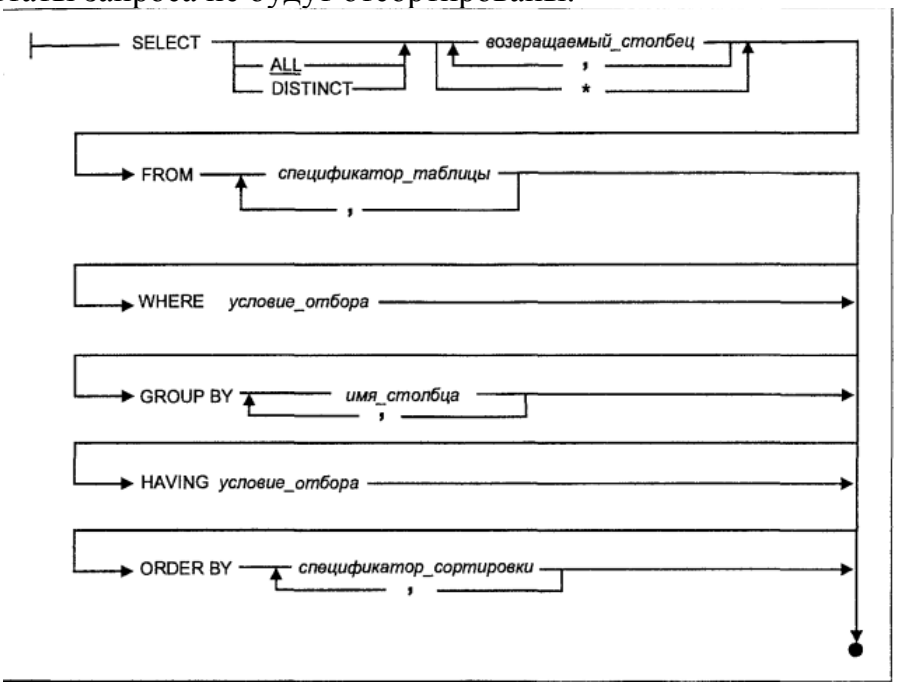

Рис. 1. Синтаксическая диаграмма инструкции SELECT

Предложение SELECT

В предложении SELECT, с которого начинаются все инструкции SELECT, **необходимо указать элементы данных, которые будут возвращены в результате запроса**. Эти элементы задаются в виде ***списка возвращаемых столбцов,*** разделенных запятыми. Для каждого элемента из этого списка в таблице результатов запроса будет создан один столбец. Столбцы в таблице результатов будут расположены в том же порядке, что и элементы списка возвращаемых столбцов. Возвращаемый столбец может представлять собой:

- *имя столбца,* идентифицирующее один из столбцов, содержащихся в таблицах, которые перечислены в предложении FROM. СУБД просто берет значе-

ние этого столбца для каждой из строк исходной таблицы и помещает его в соответствующую строку таблицы результатов запроса

- *константу,* показывающую, что в каждой строке результатов запроса должно содержаться одно и то же значение;
- *выражение,* показывающее, что СУБД должна вычислять значение, помещаемое в таблицу результатов запроса, по формуле, определенной в выражении.

Предложение FROM

Предложение **FROM** состоит из ключевого слова FROM, за которым следует *списком спецификаторов таблиц*, разделенных запятыми. Каждый спецификатор таблицы идентифицирует таблицу, содержащую данные, которые извлекает запрос. Такие таблицы называются *исходными таблицами* запроса (и инструкции SELECT), поскольку все данные, содержащиеся в таблице результатов запроса, берутся из них.

Результаты запроса на выборку

Результатом SQL-запроса на выборку всегда является таблица, содержащая данные и ничем не отличающаяся от таблиц базы данных. Если пользователь набирает инструкцию SQL в интерактивном режиме, СУБД выводит результаты запроса на экран в табличной форме. Если программа посылает запрос СУБД с помощью программного SQL, то СУБД возвращает таблицу результатов запроса программе. В любом случае результаты запроса всегда имеют такой же формат, как и обычные таблицы, содержащиеся в базе данных. Как правило, результаты запроса представляют собой таблицу с несколькими строками и столбцами. Например, запрос, приведенный ниже, возвращает таблицу из двух столбцов (поскольку список возвращаемых столбцов состоит из двух элементов) и пяти строк (поскольку количество предметов равно пяти)

Вывести список всех предметов с описанием:

```
Select [Наименование предмета], [Описание предмета]
  From Предметы
```

```
Наименование предмета                          Описание предмета
------------------------------ ----------------------------------------
Операционные системы                           Microsoft Windows Vista
Офисные пакеты                                 Microsoft Office 2007
Базы данных                                    Microsoft Access 2007
Языки программирования                         Microsoft Visual Studio 2008
Проектирование информационных систем           Microsoft SQL Server 2008
```

В отличие от запроса, показанного выше, следующий запрос возвращает одну строку, так как есть всего один студент, имеющий указанный идентификатор. Хотя результаты этого запроса, содержащие всего одну строку, имеют не такой "табличный" вид, как результаты, содержащие несколько строк, SQL все равно считает их таблицей, состоящей из трех столбцов и одной строки:

Как зовут, какой пол и дата рождения студента с идентификатором 5.

```
Select ФИО, Пол, [Дата рождения]
 From Студенты
Where [Код студента]=5
```

```
    ФИО                 Пол        Дата рождения
-------------------------------------------------- -----
Кожевников А.А.       Мужской    1981-12-04 00:00:00.000
```

В некоторых случаях результатом запроса может быть единственное значение, как в следующем примере:

Какова максимальная оценка за первый экзамен?

```
Select MAX ([Оценка 1])as [Максимальная оценка по первому экзамену]
  From Оценки
```

```
Максимальная оценка по первому экзамену
---------------------------------------
      5
```

Эти результаты запроса также считаются таблицей, которая состоит из одного столбца и одной строки.

И наконец, запрос может вернуть результаты, содержащие *нуль* строк, как в следующем примере.

Вывести название предмета, у которого описание «Web–дизайн».

```
Select [Наименование предмета]
  From Предметы
Where [Описание предмета]='Web-дизайн'
```

```
Наименование предмета
---------------------
```

Даже в таком случае результаты запроса считаются таблицей. Таблица, приведенная выше, содержит два столбца и ноль строк.

Обратите внимание, что поддержка отсутствующих данных в SQL распространяется и на результаты запроса. Если один из элементов данных в исходной таблице имеет значение NULL, то оно попадет в результаты запроса при извлечении этого элемента.

То, что SQL-запрос всегда возвращает таблицу, очень важно. Это означает, что результаты запроса можно записать обратно в базу данных в виде таблицы. Это означает также, что результаты двух запросов, имеющие похожую структуру, можно объединить в одну таблицу. И наконец, это говорит о том, что результаты запроса сами могут стать предметом дальнейших запросов. Таким образом, табличная структура реляционной базы данных тесно связана с реляционными запросами SQL. Таблицам можно посылать запросы, а запросы возвращают таблицы.

Простые запросы

Наиболее простые запросы извлекают данные из столбцов, расположенных в одной таблице базы данных. Например, следующий запрос извлекает из таблицы СТУДЕНТЫ четыре столбца:

Вывести для каждого студента ФИО, телефон, адрес и дату рождения.

```
Select ФИО, Телефон, Адрес,[Дата рождения]
  From Студенты
```

ФИО	Телефон	Адрес	Дата рождения
Иванов А.И	+74957895674	Москва	1983-12-12 00:00:00.000
Петрова И.И	+74957889876	Москва	1982-11-01 00:00:00.000
Мухин М.А.	+78462875690	Самара	1982-05-14 00:00:00.000
Сидорова В.К.	+79027868909	Саратов	1981-09-27 00:00:00.000
Кожевников А.А.	+79168563476	Казань	1981-12-04 00:00:00.000
Пальчикова Н.Е	+74569098723	Челябинск	1983-09-02 00:00:00.000
Царегородцев Е.В.	+78462234769	Самара	1980-02-17 00:00:00.000
Баранова Г.В.	+79027874638	Чебоксы	1980-07-09 00:00:00.000
Леухин П.Г.	+79067453678	Казань	1979-02-26 00:00:00.000
Николаева А.П.	+78546456432	Саратов	1979-03-17 00:00:00.000

Инструкция SELECT для простых запросов (таких, как показанных выше) состоит только из двух обязательных предложений. В предложении SELECT перечисляются имена требуемых столбцов; в предложении FROM указывается таблица, содержащая эти столбцы.

На логическом уровне запрос выполняется путем построчного просмотра таблицы, указанной в предложении FROM. Для каждой строки таблицы берутся значения столбцов, входящих в список возвращаемых столбцов, и создается одна строка результатов запроса. Таким образом, таблица результатов простого запроса на выборку содержит одну строку для каждой строки исходной таблицы базы данных.

Вычисляемые столбцы

Кроме столбцов, значения которых извлекаются непосредственно из базы данных, SQL-запрос на выборку может содержать *вычисляемые столбцы,* значения которых определяются на основании значений, хранящихся в базе данных. Чтобы получить вычисляемый столбец, в списке возвращаемых столбцов необходимо указать выражение. Выражения могут включать в себя операции сложения, вычитания, умножения и деления. В выражениях можно также использовать скобки. Конечно, столбцы, участвующие в арифметическом выражении, должны содержать числовые данные. При попытке сложить, вычесть, умножить или разделить столбцы, содержащие текстовые данные, будет выдано сообщение об ошибке.

В следующем запросе будет получен простой вычисляемый столбец.

Выдать для каждого студента оценку по первому экзамену, оценку по второму экзамену и среднюю оценку по двум экзаменам.

```
      Select [Код студента], [Оценка 1], [Оценка 2],([Оценка 2]+[Оценка 1])/2
as Средняя
        From Оценки
```

Код студента	Оценка 1	Оценка 2	Средняя
1	5	3	4
2	4	5	4
3	5	3	4
4	3	4	3
5	4	4	4
6	4	5	4
7	2	2	2
8	3	3	3
9	5	5	5
10	4	4	4

При выполнении этого запроса для каждой строки таблицы ОЦЕНКИ создается одна строка результатов. Значения первых трех столбцов результатов запроса извлекаются непосредственно из таблицы ОЦЕНКИ. Четвертый столбец для каждой строки результатов запроса вычисляется на основании значений столбцов текущей строки таблицы ОЦЕНКИ.

Во многих СУБД реализованы дополнительные арифметические операции, операции над строками символов и встроенные функции, которые можно применять в выражениях SQL. Их также можно использовать в выражениях в списке возвращаемых столбцов, как в следующем примере:

Вывести список студентов с указанием месяца рождения и года рождения.

```
Select ФИО, Month([Дата рождения])As Месяц, YEAR([Дата рождения]) As Год
  From Студенты
```

```
ФИО                                Месяц       Год
-------------------------------------------------
Иванов А.И                         12          1983
Петрова И.И                        11          1982
Мухин М.А.                         5           1982
Сидорова В.К.                      9           1981
Кожевников А.А.                    12          1981
Пальчикова Н.Е                     9           1983
Царегородцев Е.В.                  2           1980
Баранова Г.В.                      7           1980
Леухин П.Г.                        2           1979
Николаева А.П.                     3           1979
```

Кроме того, в списке возвращаемых столбцов можно использовать константы. Это может пригодиться для создания таблицы результатов запроса, которая более удобна для восприятия, как в следующем примере:

Вывести список студентов с указанием даты рождения.

```
Select ФИО, 'дата рождения',[Дата рождения]
  From Студенты
```

```
ФИО                                                  Дата рождения
------------------------------ ------------ -------------------------------
Иванов А.И                                  дата рождения 1983-12-12 00:00:00.000
Петрова И.И                                 дата рождения 1982-11-01 00:00:00.000
Мухин М.А.                                  дата рождения 1982-05-14 00:00:00.000
Сидорова В.К.                               дата рождения 1981-09-27 00:00:00.000
Кожевников А.А.                             дата рождения 1981-12-04 00:00:00.000
Пальчикова Н.Е                              дата рождения 1983-09-02 00:00:00.000
Царегородцев Е.В.                           дата рождения 1980-02-17 00:00:00.000
Баранова Г.В.                               дата рождения 1980-07-09 00:00:00.000
Леухин П.Г.                                 дата рождения 1979-02-26 00:00:00.000
Николаева А.П.                              дата рождения 1979-03-17 00:00:00.000
```

Кажется, что результаты запроса состоят из отдельных предложений, каждое из которых относится к одному из офисов, но на самом деле они представляют собой таблицу, содержащую три столбца. Первый и третий столбцы содержат значения из таблицы СТУДЕНТЫ. Во втором столбце для всех строк содержится одна и та же текстовая строка из тринадцати символов.

Выборка всех столбцов (инструкция SELECT *)

Иногда требуется получить содержимое всех столбцов таблицы. На практике такая ситуация может возникнуть, когда вы впервые сталкиваетесь с новой базой данных и необходимо быстро получить представление о ее структуре и хранимых в ней данных. С учетом этого в SQL разрешается использовать вместо списка возвращаемых столбцов символ звездочки **(*)**, который означает, что требуется извлечь все столбцы:

Показать список всех специальностей.

```
Select *
  From Специальности
```

```
Код специальности    Наименование специальности     Описание специальности
-------------------- ------------------------------------------------- -
1                    ММ                             Математические методы
2                    ПИ                             Прикладная информатика
3                    СТ                             Статистика
4                    МО                             Менеджмент организаций
5                    БУ                             Бухгалтерский учет
```

Результаты запроса содержат все пять столбцов таблицы СПЕЦИАЛЬНОСТИ, которые расположены в том же порядке, что и в исходной таблице.

В стандарте ANSI/ISO сказано, что в предложении SELECT можно использовать либо символ выборки всех столбцов, либо список возвращаемых столбцов, но не оба одновременно. Однако во многих СУБД символ звездочки считается просто одним из возвращаемых столбцов. Таким образом, запрос

```
Select *, [Оценка 1]-[Оценка 2]
  From Оценки
```

допустим в большинстве коммерческих диалектов SQL (в частности, в DB2, Oracle и SQL Server), однако не разрешен стандартом ANSI/ISO.

Символ выборки всех столбцов очень удобно использовать в интерактивном SQL. В то же время, следует избегать использования его в программном SQL, поскольку изменения в структуре базы данных могут привести к краху приложения. Предположим, например, что таблица ОЦЕНКИ была удалена из базы данных, а затем создана вновь, при этом был изменен порядок столбцов и добавлен новый, четвертый столбец. Если программа ожидает, что запрос SELECT * FROM ПРЕДМЕТЫ возвратит таблицу, содержащую три столбцов определенных типов, она почти наверняка перестанет работать после изменения порядка столбцов и добавления нового столбца.

Этих сложностей можно избежать, если в программах запрашивать требуемые столбцы по именам. Например, следующий запрос возвращает те же результаты, что и запрос SELECT * FROM ПРЕДМЕТЫ. Он не восприимчив к изменениям структуры базы данных, пока в таблице ПРЕДМЕТЫ существуют столбцы с указанными именами.

```
Select [Код предмета], [Наименование предмета], [Описание предмета]
  From Предметы
```

Повторяющиеся строки (предикат DISTINCT)

Если в списке возвращаемых столбцов запроса на выборку указать первичный ключ таблицы, то каждая строка результатов запроса будет уникальной (из-за того, что значения первичного ключа во всех строках разные). Если первичный ключ не указан, результаты запроса могут содержать повторяющиеся строки. Предположим, например, что был выполнен следующий запрос:

Вывести список всех адресов студентов

```
Select адрес
  From студенты
```

```
адрес
--------
Москва
Москва
Самара
Саратов
Челябинск
Чебоксары
Казань
Самара
Саратов
```

Таблица результатов запроса содержит девять строк (по одной для каждого офиса), однако четыре пары из них совпадают. Это не совсем те результаты, которых вы ожидали. Если в фирме работают четыре менеджера, то, вероятнее всего, вы ожидали, что результаты запроса будут содержать шесть строк.

Повторяющиеся строки из таблицы результатов запроса можно удалить, если в инструкции SELECT перед списком возвращаемых столбцов указать предикат DISTINCT

Вывести список всех адресов студентов

```
Select DISTINCT адрес
  From студенты
```

```
адрес
--------
Казань
Москва
Самара
Саратов
Челябинск
Чебоксары
```

Этот запрос выполняется следующим образом. Вначале генерируются все строки таблицы результатов (восемь строк), а затем удаляются те из них, которые в точности повторяют другие. Предикат DISTINCT можно указывать независимо от содержимого списка возвращаемых столбцов инструкции SELECT.

Если предикат DISTINCT не указан, повторяющиеся строки не удаляются. Можно также задать предикат ALL, явно показывая, что повторяющиеся строки следует оставить, однако делать это не обязательно, поскольку предикат ALL используется по умолчанию.

Отбор строк (предложение WHERE)

SQL-запросы, извлекающие из таблицы все строки, полезны при просмотре базы данных и создании отчетов, однако редко применяются для чего-нибудь еще.

Обычно требуется выбрать из таблицы несколько строк и включить в результаты запроса только их. Чтобы указать, какие строки нужно отобрать, следует воспользоваться предложением WHERE.

Вывести ФИО, пол, возраст студента, у которого код равен 1

```
Select ФИО, пол, [количество лет при поступлении]
  From студенты
  Where [код студента]=1
```

```
ФИО              пол        количество лет при поступлении
---------------- ---------- ------------------------------
Иванов А.И.      мужской              24
```

Предложение WHERE состоит из ключевого слова WHERE, за которым следует условие отбора, определяющее, какие именно строки требуется извлечь. Если в условии отбора встречается имя столбца (как, например, имя КОД СТУДЕНТА в примере), то используется значение этого столбца из текущей строки. Для каждой из строк условие отбора может иметь одно из трех перечисленных ниже значений.

- Если условие отбора имеет значение TRUE, строка будет включена в результаты запроса.
- Если условие отбора имеет значение FALSE, то строка исключается из результатов запроса.
- Если условие отбора имеет значение NULL, то строка исключается из результатов запроса.

Можно сказать, что условие отбора служит фильтром для строк таблицы. Строки, удовлетворяющие условию отбора, проходят через фильтр и становятся частью результатов запроса. Строки, не удовлетворяющие условию отбора, отфильтровываются и исключаются из результатов запроса.

Условия отбора

В SQL используется множество условий отбора, позволяющих эффективно и естественным образом создавать различные типы запросов. Ниже рассматриваются пять основных условий отбора (в стандарте ANSI/ISO они называются *предикатами):*

1. *Сравнение.* Значение одного выражения сравнивается со значением другого выражения. Например, такое условие отбора используется для выбора всех офисов, находящихся в восточном регионе, или всех служащих, фактические объемы продаж которых превышают плановые.
2. *Проверка на принадлежность диапазону.* Проверяется, попадает ли указанное значение в определенный диапазон. Например, такое условие отбора используется для нахождения служащих, фактические объемы продаж которых превышают $100000, но меньше $500000.
3. *Проверка на членство в множестве.* Проверяется, совпадает ли значение выражения с одним из значений заданного множества. Например, такое условие отбора используется для выбора студентов, проживающих в Москве, Саратове или Казани.
4. *Проверка на соответствие шаблону.* Проверяется, соответствует ли строковое значение, содержащееся в столбце, определенному шаблону. Например,

такое условие отбора используется для выбора студентов, фамилии которых начинаются с буквы "И".

5. *Проверка на равенство значению NULL*. Проверяется, содержится ли в столбце значение NULL. Например, такое условие отбора используется для нахождения всех студентов, которые еще не сдавали ни одного экзамена.

Сравнение (операторы =, <>, <, <=, >, >=)

Наиболее распространенным условием отбора в SQL является *сравнение.* При сравнении СУБД вычисляет и сравнивает значения двух выражений для каждой строки данных. Выражения могут быть как очень простыми, например, содержать одно имя столбца или константу, так и более сложными, например, содержать арифметические операции. В SQL имеется шесть различных способов сравнения двух выражений, как изображено на рисунке 2.

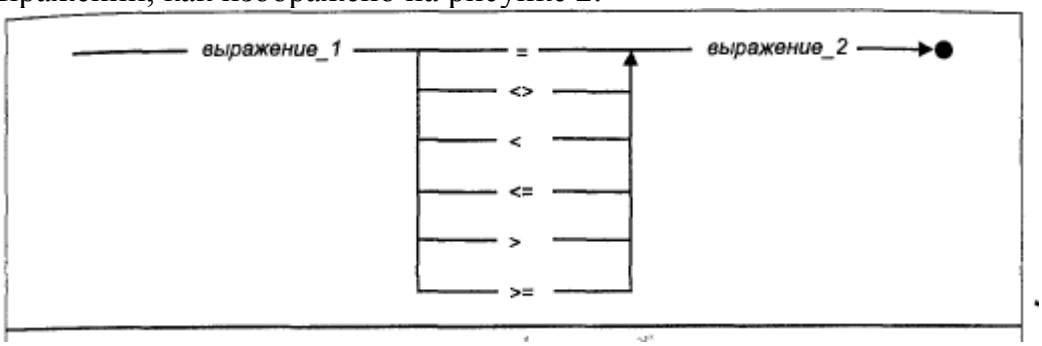

Рис. 2. Синтаксическая диаграмма операции сравнения

Найти всех студентов, которые родились до 1980 года

```
Select  ФИО, [дата рождения]
  From студенты
  Where [дата рождения]> '01/01/1980'
```

```
  ФИО                      дата рождения
---------------------- -------------------
Иванов А.И.          1983-12-12 00:00:00.000
Петрова И.И.         1982-11-01 00:00:00.000
Мухин М.А.           1982-05-14 00:00:00.000
Сидорова В.К.        1981-09-27 00:00:00.000
Пальчикова Н.Е.      1981-09-02 00:00:00.000
Царегородцев Е.В.    1980-02-17 00:00:00.000
Баранова Г.В.        1980-07-09 00:00:00.000
```

Вывести список всех студентов, у которых курс <>4

```
Select  ФИО, курс
  From студенты
  Where курс <>4
```

```
ФИО                         курс
--------------------------- ----
Иванов А.И.                  1
Петрова И.И.                 2
Мухин М.А.                   2
Сидорова В.К.                3
Пальчикова Н.Е.              1
Леухин П.Г.                  5
```

Николаева А.П. 5

Как видно из рис. 2, в соответствии со спецификацией ANSI/ISO проверка на неравенство записывается как "A <> B". В ряде СУБД используются альтернативные системы записи, как, например, "A != B" (в SQL Server) и "A != B" (в DB2 и SQL/DS). Иногда такая форма записи является единственной, а иногда — только одной из допустимых форм.

Когда СУБД сравнивает значения двух выражений, могут получиться три результата:

- если сравнение истинно, то результат проверки имеет значение TRUE;
- если сравнение ложно, то результат проверки имеет значение FALSE;
- если хотя бы одно из двух выражений имеет значение NULL, то результатом проверки будет NULL

Выборка одной строки

Чаще всего используется сравнение, в котором определяется, равно ли значение столбца некоторой константе. Если этот столбец представляет собой первичный ключ, то запрос возвращает всего одну строку.

Вывести ФИО, пол, возраст студента, у которого код равен 1

```
Select ФИО, пол, [количество лет при поступлении]
  From студенты
  Where [код студента]=1
```

```
ФИО              пол        количество лет при поступлении
---------------- ---------- ------------------------------
Иванов А.И.      мужской              24
```

Запросы этого типа часто применяются в программах просмотра баз данных, построенных на основе форм. Пользователь вводит в форму идентификатор клиента, и программа использует его при создании и выполнении запроса. После этого она отображает извлеченные данные в форме.

Замечания по поводу значений NULL. При определении условия отбора необходимо помнить об обработке значений NULL. В трехзначной логике, принятой в SQL, условие отбора может иметь значения TRUE, FALSE или NULL. А в результаты запроса попадают только те строки, для которых условие отбора равно TRUE.

Проверка на принадлежность диапазону значений (оператор BETWEEN...AND)

Другой формой условия отбора является проверка на принадлежность диапазону значений (оператор BETWEEN. . .AND), схематически изображенная на рис. 3. При этом проверяется, находится ли элемент данных между двумя заданными значениями. В условие отбора входят три выражения. Первое выражение определяет проверяемое значение; второе и третье выражения определяют верхний и нижний пределы проверяемого диапазона. Типы данных трех выражений должны быть сравнимыми.

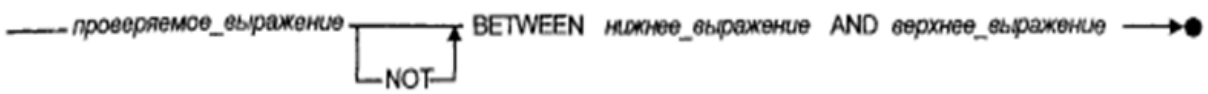

Рис. 3. Синтаксическая диаграмма проверки на принадлежность диапазону (оператор BETWEEN...AND)

Найти всех студентов, которые родились с 1979 по 1981 год

```
Select  ФИО, [дата рождения]
  From студенты
  Where [дата рождения] between '01.01.1979' and '01.01.1981'
```

```
ФИО                                                  дата рождения
---------------------------------------------------- -----------------------
Царегородцев Е.В.                                    1980-02-17 00:00:00.000
Баранова Г.В.                                        1980-07-09 00:00:00.000
Леухин П.Г.                                          1979-02-26 00:00:00.000
Николаева А.П.                                       1979-03-17 00:00:00.000
```

При проверке на принадлежность диапазону верхний и нижний пределы считаются частью диапазона.

Инвертированная версия проверки на принадлежность диапазону позволяет выбрать значения, которые лежат за пределами диапазона, как в следующем примере:

Найти всех студентов, которые не родились с 1979 по 1981 год

```
Select  ФИО, [дата рождения]
  From студенты
  Where  [дата  рождения]  not  between  '01.01.1979'  and  '01.01.1981'
```

```
ФИО                                                  дата рождения
---------------------------------------------------- -----------------------
Иванов А.И.                                          1983-12-12 00:00:00.000
Петрова И.И.                                         1982-11-01 00:00:00.000
Мухин М.А.                                           1982-05-14 00:00:00.000
Сидорова В.К.                                        1981-09-27 00:00:00.000
Пальчикова Н.Е.                                      1981-09-02 00:00:00.000
```

Поверяемое выражение, заданное в операторе `BETWEEN. . .AND`, может быть любым допустимым выражением, однако на практике оно обычно представляет собой короткое имя столбца.

В стандарте ANSI/ISO определены относительно сложные правила обработки значений `NULL` в проверке `BETWEEN`:

- если проверяемое выражение имеет значение NULL либо если оба выражения, определяющие диапазон, равны NULL, то проверка BETWEEN возвращает NULL;
- если выражение, определяющее нижнюю границу диапазона, имеет значение NULL, то проверка BETWEEN возвращает FALSE, когда проверяемое значение больше верхней границы диапазона, и NULL в противном случае;
- если выражение, определяющее верхнюю границу диапазона, имеет значение NULL, то проверка BETWEEN возвращает FALSE, когда проверяемое значение меньше нижней границы диапазона, и NULL в противном случае.

Однако прежде чем полагаться на эти правила, неплохо было бы поэкспериментировать со своей СУБД.

Необходимо отметить, что проверка на принадлежность диапазону не расширяет возможности SQL, поскольку ее можно выразить в виде двух сравнений. Проверка

A BETWEEN B AND C

полностью эквивалентна следующему сравнению:

(A >= B) AND (A <= C)

Тем не менее, проверка BETWEEN является более простым способом выразить условие отбора в терминах диапазона значений.

Поверка на членство в множестве (оператор IN)

Еще одним распространенным условием отбора является проверка на членство в множестве (оператор `IN`), схематически изображенная на рис. 4. В этом случае проверяется, соответствует ли элемент данных какому-либо значению из заданного списка.

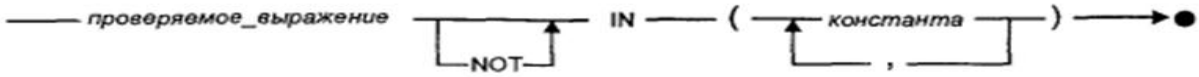

Рис. 4. Синтаксическая диаграмма проверки на членство в множестве (оператор IN)

Вывести список студентов, которые живут в Самаре, Казани и Москве

```
Select  ФИО, адрес
  From студенты
  Where адрес in ('Москва','Саратов','Казань')
```

```
ФИО                                                адрес
-------------------------------------------------- ------
Иванов А.И.                                        Москва
Петрова И.И.                                       Москва
Мухин М.А.                                         Самара
Баранова Г.В.                                      Казань
Леухин П.Г.                                        Самара
```

С помощью проверки NOT IN можно убедиться в том, что элемент данных *не* является членом заданного множества. Проверяемое выражение в операторе IN может быть любым допустимым выражением, однако обычно оно представляет собой короткое имя столбца, как в предыдущих примерах. Если результатом проверяемого выражения является значение NULL, то проверка IN также возвращает NULL. Все элементы в списке заданных значений должны иметь один и тот же тип данных, который должен быть сравним с типом данных проверяемого выражения.

Как и проверка BETWEEN, проверка IN не добавляет в возможности SQL ничего нового, поскольку условие

X IN (A, B, C)

полностью эквивалентно условию

(X = A) OR (X = B) OR (X = C)

Однако проверка IN предлагает гораздо более эффективный способ выражения условия отбора, особенно если множество содержит большое число элементов. В стандарте ANSI/ISO не определено максимальное количество элементов множества, и в большинстве СУБД не задан явно верхний предел. По соображениям пе-

реносимости лучше избегать множеств, содержащих один элемент, наподобие такого:

CITY IN ('Москва')

Их следует заменять следующим простым сравнением:

CITY = 'Москва'

Проверка на соответствие шаблону (оператор LIKE)

Для выборки строк, в которых содержимое некоторого текстового столбца совпадает с заданным текстом, можно использовать простое сравнение.

Показать наименование специальности, у которой описание Статистика

```
Select  [название специальности],[ описание специальности]
  From специальности
  Where [описание специальности]='Статистика'
```

```
название специальности                              описание специальности
-------------------------------------------------- ----------------------
СТ                                                 Статистика
```

Однако очень легко можно забыть, какое именно название носит интересующая нас специальность: "Статистика" или "Статика". Проверка на соответствие шаблону позволяет выбрать из базы данных строки на основе частичного соответствия имени специальности.

Поверка на соответствие шаблону (оператор `LIKE`), схематически изображенная на рис. 5, позволяет определить, соответствует ли значение данных в столбце некоторому *шаблону*. Шаблон представляет собой строку, в которую может входить один или более *подстановочных знаков*. Эти знаки интерпретируются особым образом.

Рис. 5. Синтаксическая диаграмма проверки на соответствие шаблону (оператор LIKE)

Подстановочные знаки

Подстановочный знак % совпадает с любой последовательностью из нуля или более символов. Оператор like указывает СУБД, что необходимо сравнивать содержимое столбца с каким-нибудь шаблоном, например, "Smith% Corp.". Этому шаблону соответствуют все перечисленные имена.

Smith Corp. Smithson Corp. Smithsen Corp. Smithsonian Corp.

А вот следующие имена данному шаблону не соответствуют

SmithCorp Smithson Inc.

Подстановочный знак _ (символ подчеркивания) совпадает с любым отдельным символом. Например, если вы уверены, что название компании — либо "Smithson", либо "Smithsen", то можете воспользоваться следующей конструкцией: LIKE 'Smithsen Corp.'

В таком случае любое из этих имен будет соответствовать шаблону

Smithson Corp. Smithsen Corp. Smithsun Corp.

а ни одно из этих не будет ему соответствовать:

Smithsoon Corp. Smithsn Corp.

Подстановочные знаки можно помещать в любое место строки шаблона, и в одной строке может содержаться несколько подстановочных знаков. Следующий конструкция допускает как написание "Smithson" и "Smithsen", так и любое другое окончание названия компании, включая "Corp.", "Inc." или какое-то другое: LIKE 'Smiths_n %'

С помощью формы not like можно находить строки, которые не соответствуют шаблону. Проверку like можно применять только к столбцам, имеющим строковый тип данных. Если в столбце содержится значение null, то результатом проверки like ,будет null.

Вероятно, приходилось уже встречаться с проверкой на соответствие шаблону в операционных системах, имеющих интерфейс командной строки (Unix, MS-DOS). В этих системах звездочка (*) используется для тех же целей, что и символ процента (%) в SQL, а вопросительный знак (?) соответствует символу подчеркивания (_) в SQL, но в целом возможности работы с шаблонами строк в них такие же.

Символы пропуска

При проверке строк на соответствие шаблону может оказаться, что подстановочные знаки входят в строку символов в качестве литералов. Например, нельзя проверить, содержится ли знак процента в строке, просто включив его в шаблон, поскольку СУБД будет считать этот знак подстановочным. Как правило, это не вызывает серьезных проблем, поскольку подстановочные знаки довольно редко встречаются в именах, названиях товаров и других текстовых данных, которые обычно хранятся в базе данных.

Проверка на равенство значению NULL (оператор IS NULL)

Значения NULL обеспечивают возможность применения трехзначной логики в условиях отбора. Для любой заданной строки результат применения условия отбора может быть TRUE, FALSE ИЛИ NULL (В случае, когда в одном из столбцов содержится значение NULL) Иногда бывает необходимо явно проверять значения столбцов на равенство NULL И непосредственно обрабатывать их. Для этого в SQL имеется специальная проверка is NULL, синтаксическая диаграмма которой изображена на рис. 6.

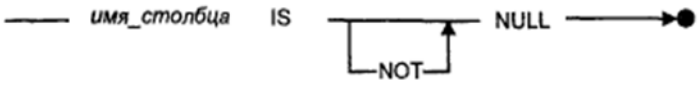

Рис. 6 Синстаксическая диаграмма проверки на равенство значению NULL (оператор IS NULL)

Инвертированная форма проверки на NULL (IS NOT NULL) позволяет отыскать строки, которые не содержат значений NULL.

В отличие от условий отбора, описанных выше, проверка на NULL не может возвратить значение NULL в качестве результата. Она всегда возвращает TRUE или FALSE.

Может показаться странным, что нельзя проверить значение на равенство NULL с помощью операции сравнения.

Ключевое слово null здесь нельзя использовать, поскольку на самом деле это не настоящее значение; это просто сигнал о том, что значение неизвестно. Даже если бы сравнение

Курс = NULL

было возможно, правила обработки значений NULL в сравнениях привели бы к тому, что оно вело бы себя не так, как ожидается. Если бы СУБД обнаружила строку, в которой столбец REP_OFFICE содержит значение NULL, выполнилась бы следующая проверка:

NULL = NULL

Что будет результатом этого сравнения: true или false? Так как значения по обе стороны знака равенства неизвестны, то, в соответствии с правилами логики SQL, условие отбора должно вернуть значение null. Поскольку условие отбора возвращает результат, отличный от true, строка исключается из таблицы результатов запроса — это противоположно тому, к чему вы стремились! Из-за правил обработки значений null в SQL необходимо использовать проверку is null.

Составные условия отбора (операторы AND, OR и NOT)

Простые условия отбора, описанные в предыдущих параграфах, после применения к некоторой строке возвращают значения true, false или null. С помощью правил логики эти простые условия можно объединять в более сложные, как изображено на рис. 7.

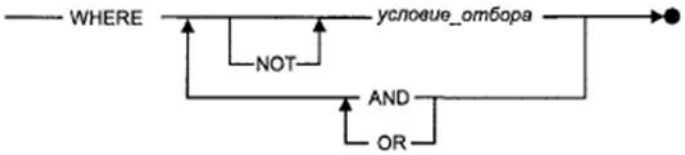

Рис. 7 Синтаксическая диаграмма составных условий отбора (операторы AND, OR и NOT)

Условия отбора, объединенные с помощью операторов AND, OR И NOT, сами могут быть составными.

Оператор OR используется для объединения двух условий отбора, из которых как минимум одно должно быть истинным:

Найти все студентов 2 или 4 курса

```
SELECT      ФИО, курс
  FROM          студенты
  WHERE      (курс = 2) OR
             (курс = 4)
```

```
ФИО                                                 курс
--------------------------------------------------- ----
Петрова И.И.                                        2
Мухин М.А.                                          2
Царегородцев Е.В.                                   4
Баранова Г.В.                                       4
```

Для объединения двух условий отбора, оба из которых должны быть истинными, следует использовать оператор AND:

Найти всех студентов, которые родились раньше 1980 и учатся на 5 курсе

```
SELECT     ФИО, [дата рождения], курс
  FROM         студенты
  WHERE     ([дата рождения] < '01.01.1980') AND (курс = 5)
```

```
ФИО                                                дата рождения          курс
-------------------------------------------------- -------------------------
Леухин П.Г.                                        1979-02-26 00:00:00.000 5
Николаева А.П.                                     1979-03-17 00:00:00.000 5
```

И наконец, можно использовать оператор NOT, чтобы выбрать строки, для которых условие отбора ложно:

Найти студентов, год рождения которых позже 1980, но курс не меньше 4-го

```
SELECT     ФИО, [дата рождения], курс
  FROM         студенты
  WHERE     ([дата рождения] < '01.01.1980') AND (NOT (курс < 4))
```

```
ФИО                                                дата рождения          курс
-------------------------------------------------- -------------------------
Леухин П.Г.                                        1979-02-26 00:00:00.000 5
Николаева А.П.                                     1979-03-17 00:00:00.000 5
```

С помощью логических операций AND, OR и NOT и круглых скобок можно создавать очень сложные условия отбора:

Найти студентов, которые:

a) *имеют женский пол*
b) *и которые родились раньше 1981 года*
c) *но не которые учатся на дневной форме обучения*

```
SELECT     ФИО, [дата рождения], курс, пол, [очная форма обучения]
  FROM         студенты
  WHERE      ([дата рождения] < '01.01.1981') AND (пол = 'женский') AND
(NOT ([очная форма обучения] = 1))
```

```
ФИО                  дата рождения           курс   пол   очная форма обучения
-------------------------------------------------- ---------------------
Баранова Г.В.       1980-07-09 00:00:00.000   4    женский    0
Николаева А.П.      1979-03-17 00:00:00.000   5    женский    0
```

Как и в случае с простыми условиями отбора, значения NULL влияют на интерпретацию составных условий отбора, вследствие чего результаты последних становятся не столь очевидными. В частности, результатом операции NULL OR TRUE является значение TRUE, а не NULL, как можно было ожидать. Таблицы 1, 2 и 3 являются таблицами истинности для операторов AND, OR И NOT соответственно; здесь же показано влияние значений NULL.

Таблица 1. Таблица истинности проверки оператора AND

AND	TRUE	FALSE	NULL
TRUE	TRUE	FALSE	NULL
FALSE	FALSE	FALSE	FALSE
NOLL	NULL	FALSE	NULL

Таблица 2. Таблица истинности оператора OR

OR	TRUE	FALSE	NULL
TRUE	TRUE	TRUE	TRUE
FALSE	TRUE	FALSE	NULL
NULL	TRUE	NULL	NULL

Таблица 3 Таблица истинности оператора N0T

NOT	TRUE	FALSE	NULL
	FALSE	TRUE	NULL

Сортировка результатов запроса (предложение ORDER BY)

Строки результатов запроса, как и строки таблицы базы данных, не имеют определенного порядка. Но, включив в инструкцию SELECT предложение ORDER BY, можно отсортировать результаты запроса. Это предложение, синтаксическая диаграмма которого изображена на рис. 8, содержит список имен или порядковых номеров столбцов, разделенных запятыми.

Вывести всех студентов, отсортированных в алфавитном порядке по ФИО

```
SELECT     ФИО
  FROM         студенты
  ORDER BY ФИО
```

```
ФИО
--------------------------------------------------
Баранова Г.В.
Иванов А.И.
Леухин П.Г.
Мухин М.А.
Николаева А.П.
Пальчикова Н.Е.
Петрова И.И.
Сидорова В.К.
Царегородцев Е.В.
```

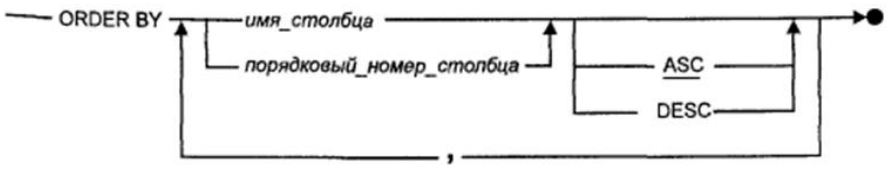

Рис. 8 Сортировка результатов запроса (предложение ORDER BY)

Можно сортировать результаты запроса по любому элементу списка возвращаемых столбцов.

В предложении ORDER BY можно выбрать возрастающий или убывающий порядок сортировки. По умолчанию данные сортируются в порядке возрастания.

Чтобы сортировать их по убыванию, следует включить в предложение сортировки ключевое слово DESC, как это сделано в следующем примере:

Вывести студентов, курс обучения которых отсортирован в порядке убывания

```
SELECT      ФИО, курс
  FROM         студенты
  ORDER BY курс DESC
```

```
ФИО                                                курс
-------------------------------------------------- ----
Леухин П.Г.                                        5
Николаева А.П.                                     5
Царегородцев Е.В.                                  4
Баранова Г.В.                                      4
Сидорова В.К.                                      3
Петрова И.И.                                       2
Мухин М.А.                                         2
Пальчикова Н.Е.                                    1
Иванов А.И.                                        1
```

Как свидетельствует рис. 8, чтобы определить порядок сортировки по возрастанию, необходимо использовать ключевое слово ASC, однако из-за того что такой порядок принят по умолчанию, это ключевое слово обычно не указывают.

Если столбец результатов запроса, используемый для сортировки, является вычисляемым, то у него нет имени, которое можно указать в предложении сортировки. В таком случае вместо имени столбца необходимо указать его порядковый номер:

Показать список всех студентов, отсортированный по разности между датой поступления и датой рождения в порядке убывания

```
SELECT     ФИО, ([дата поступления] - [дата рождения])
  FROM         студенты
  ORDER BY 2 DESC
```

```
Пальчикова Н.Е.                                    1925-11-30 00:00:00.000
Леухин П.Г.                                        1924-05-28 00:00:00.000
Царегородцев Е.В.                                  1924-05-19 00:00:00.000
Мухин М.А.                                         1924-02-23 00:00:00.000
Баранова Г.В.                                      1924-02-02 00:00:00.000
Сидорова В.К.                                      1923-10-26 00:00:00.000
Петрова И.И.                                       1923-10-02 00:00:00.000
Иванов А.И.                                        1923-09-22 00:00:00.000
Николаева А.П.                                     1922-04-08 00:00:00.000
```

Правила выполнения однотабличных запросов

Однотабличные запросы в большинстве своем являются простыми, и смысл такого запроса обычно можно легко понять, просто прочитав инструкцию SELECT. Однако по мере возрастания сложности запроса появляется необходимость в более точном определении результатов, которые будут возвращены данной инструкцией SELECT. В таблице описана процедура генерации результатов SQL-запроса.

Результаты запроса, возвращенные инструкцией SELECT, получаются в результате поочередного применения входящих в инструкцию предложений. Вначале

применяется предложение FROM (оно выбирает таблицу, содержащую требуемые данные), затем — WHERE (которое по определенному критерию отбирает из таблицы строки), далее — SELECT (которое создает указанные столбцы результатов запроса и при необходимости удаляет повторяющиеся строки) и, наконец, ORDER BY (сортирует результаты запроса).

Таблица результатов запроса на выборку генерируется следующим образом:

1. Взять таблицу, указанную в предложении FROM.
2. Если имеется предложение WHERE, применить заданное в нем условие отбора к каждой строке таблицы и оставить только те строки, для которых это условие выполняется, т.е. имеет значение TRUE; строки, для которых условие отбора имеет значение FALSE или NULL, — отбросить.
3. Для каждой из оставшихся строк вычислить значение каждого элемента в списке возвращаемых столбцов и создать одну строку таблицы результатов запроса. При каждой ссылке на столбец используется значение столбца для текущей строки.
4. Если указано ключевое слово DISTINCT, удалить из таблицы результатов запроса все повторяющиеся строки.
5. Если имеется предложение ORDER BY, отсортировать результаты запроса.

Объединение результатов нескольких запросов (операция UNION)

Иногда появляется необходимость объединения результатов двух или более запросов в одну таблицу. SQL поддерживает такую возможность с помощью операции UNION.

Вывести список студентов, которые учатся на 2 курсе

```
SELECT     ФИО, курс
  FROM         студенты
  WHERE     (курс = 2)
```

```
ФИО                                                 курс
--------------------------------------------------- ----
Петрова И.И.                                         2
Мухин М.А.                                           2
```

Вывести список студентов, которые учатся на 4 курсе

```
SELECT     ФИО, курс
  FROM         студенты
  WHERE     (курс = 4)
```

```
ФИО                                                 курс
--------------------------------------------------- ----
Царегородцев Е.В.                                    4
Баранова Г.В.                                        4
```

Объединение запросов:

Вывести студентов, которые учатся на 2 и 4 курсе:

```
SELECT     ФИО, курс
FROM         студенты
WHERE     (курс = 2)
```

```
UNION
SELECT      ФИО, курс
FROM          студенты
WHERE      (курс = 4)
```

ФИО	курс
Баранова Г.В.	4
Мухин М.А.	2
Петрова И.И.	2
Царегородцев Е.В.	4

Чтобы таблицы результатов запроса можно было объединить с помощью операции UNION, они должны соответствовать следующим требованиям:

- две таблицы должны содержать одинаковое число столбцов;
- тип данных каждого столбца первой таблицы должен совпадать с типом данных соответствующего столбца во второй таблице;
- ни одна из двух таблиц не может быть отсортирована с помощью предложения *J* ORDER BY; однако объединенные результаты запроса можно отсортировать, как описано ниже.

Имена столбцов в двух запросах, объединенных с помощью операции UNION, не обязательно должны быть одинаковыми. Поскольку столбцы в двух таблицах могут иметь различные имена, столбцы результатов запроса, возвращенные операцией UNION, являются безымянными.

Запрос на объединение и повторяющиеся строки

По умолчанию операция UNION в процессе своего выполнения *удаляет* повторяющиеся строки.

Если в таблице результатов операции UNION необходимо сохранить повторяющиеся строки, сразу за ключевым словом UNION следует указать предикат ALL. Эта форма запроса вернет таблицу с повторяющимися строками.

Обработка повторяющихся строк в операции UNION и инструкции SELECT осуществляется по-разному. Инструкция SELECT ПО умолчанию оставляет такие строки (SELECT ALL). Чтобы удалить их, необходимо явно указать предикат DISTINCT. Операция UNION по умолчанию удаляет повторяющиеся строки. Чтобы оставить их, следует явно задать предикат ALL.

Удаление повторяющихся строк из таблицы результатов запроса занимает много времени, особенно если таблица содержит большое количество строк. Если известно, что операция UNION не возвратит повторяющихся строк, необходимо явно указать предикат ALL, тогда запрос будет выполняться быстрее.

Запрос на объединение и сортировка

Предложение ORDER BY нельзя использовать ни в одной из инструкций SELECT, объединенных операцией UNION. Нет смысла выполнять сортировку результатов таких запросов, поскольку пользователь все равно не увидит их в чистом виде. Однако *объединенные* результаты запросов, возвращенные операцией UNION, можно отсортировать с помощью предложения ORDER BY, следующего за

второй инструкцией SELECT. Поскольку столбцы таблицы результатов запроса на объединение не имеют имен, в этом предложении следует указывать номера столбцов.

Вывести студентов, которые учатся на 2 и 4 курсе, отсортировать в алфавитном порядке.

```
SELECT     ФИО, курс
  FROM         студенты
  WHERE     (курс = 2)
UNION
SELECT     ФИО, курс
  FROM         студенты
  WHERE     (курс = 4)
order by 1
```

```
ФИО                                                курс
-------------------------------------------------- ----
Баранова Г.В.                                      4
Мухин М.А.                                         2
Петрова И.И.                                       2
Царегородцев Е.В.                                  4
```

Вложенные запросы на объединение

Операцию UNION можно использовать многократно, чтобы объединить результаты трех или более запросов. В результате следующего запроса сначала происходит объединение таблиц B и C – получается одна объединенная таблица. Потом с помощью другой операции UNION эта таблица объединяется с таблицей A:

```
SELECT *
   FROM A
   UNION (SELECT *
            FROM B
          UNION
          SELECT *
             FROM C)
```

Скобки в запросе показывают, какая операция UNION должна выполняться первой Независимо от того, удаляют ли все операции UNION повторяющиеся строки или сохраняют их, порядок выполнения инструкций не имеет значения. Следующие выражения полностью эквивалентны

A UNION (B UNION C)
(A UNION B) UNION C
(A UNION C) UNION B

Подобным образом три следующих выражения полностью эквивалентны, поскольку повторяющиеся строки сохраняются:

A UNION ALL (B UNION ALL C)
(A UNION ALL B) UNION ALL C
(A UNION ALL C) UNION ALL B

Однако если в запросы на объединения входят как операции UNION, так и операции UNION ALL, то порядок следования этих инструкций имеет значение. Выражение

A UNION ALL B UNION C

можно интерпретировать 2 способами

1) A UNION ALL B UNION C
2) A UNION ALL (B UNION C)

Однако если его проинтерпретировать как

(A UNION ALL B) UNION C

то результаты этих 2-х запросов будут возвращать разное количество строк. По этой причине всегда необходимо использовать круглые скобки, чтобы указать последовательность выполнения в запросах на объединение, содержащих три или более операций UNION.

Порядок выполнения работы

1. Изучить теоретический материла.

2. Выполнить все примерами и проверить результаты выполнения запроса на соответствие данным в базе данных.

3. Получить у преподавателя задание для индивидуальной работы.

4. Ответить на контрольные вопросы.

Контрольные вопросы

1. Назовите основные функции инструкции SELECT.
2. Перечислите и опишите основные предложения инструкции SELECT.
3. Что необходимо указать в предложении SELECT? Дайте определение понятию **список возвращаемых столбцов**.
4. В каком порядке расположены столбцы в результате запроса?
5. Перечислите и опишите основные виды результатов запроса.
6. Дайте определения следующим понятиям: **исходные таблицы запроса**, **вычисляемые столбцы**
7. Какой формат имеют результаты запроса на выборку?
8. Какой символ используется для извлечения всех столбцов отношения?
9. Какой предикат необходимо указать в инструкции SELECT перед списком возвращаемых столбцов, чтобы избежать повторения строк в результате выполнения запроса?

Преподаватель С.В. Банцевич

Рассмотрено на заседании цикловой
комиссии программного обеспечения
информационных технологий №10
Протокол № от « »_____201_
Председатель ЦК Т.Г.Багласова