Частное учреждение образования
«Колледж бизнеса и права»

УТВЕРЖДАЮ
Заместитель директора
по учебной работе
_____И.В.Малафей.
«___»_____2017

Специальность: «Программное обеспечение информационных технологий»	Дисциплина: «Базы данных и системы управления базами данных»

Лабораторная работа № 8

Инструкционно-технологическая карта

Тема: Создание запросов к нескольким таблицам
Цель работы:

– познакомиться с созданием простых запросов на объединение таблиц (объединение по равенству), запросов с использованием отношения предок/потомок к двум и более таблицам;

– приобрести умения применять в многотабличных запросах на выборку полные имена столбцов, псевдонимы таблиц;

– изучить понятие связного столбца;

– получить представление об объединении таблиц по неравенству;

– изучить на примерах особенности реализации многотабличных запросов.

Время выполнения: 2 часа

Краткие теоретические сведения

Пример двухтабличного запроса

На практике многие запросы считывают информацию сразу из нескольких таблиц базы данных.

Чтобы понять, как в SQL реализуются многотабличные запросы, лучше всего начать с рассмотрения простого запроса, который объединяет данные из двух различных таблиц.

Вывести список ФИО, дату рождения и пол студентов, указав соответствующие специализации для каждого.

Четыре запрашиваемых элемента данных хранятся в двух различных таблицах:

- в таблице *Специальности* содержится названия специальности каждого студента;
- в таблице *Студенты* содержаться имена, дата рождения и пол каждого студента.

Однако между двумя этими таблицами существует связь. В каждой строке столбца *Код специальности* таблицы *Студенты* содержится идентификатор специальности. Очевидно, чтобы получить требуемые результаты, в инструкции *select*, **с** помощью которой осуществляется запрос, необходимо как-то учесть эту связь между таблицами.

При получении результатов такого запроса буду полезны две вещи:

- каждая строка таблицы результатов запроса формируется из *пары* строк: одна строка находится в таблице *Студенты*, а другая — в таблице *Специальности.*
- для нахождения данной пары строк производится сравнение содержимого *соответствующих столбцов* в этих таблицах.

Простое объединение таблиц (объединение по равенству)

Процесс формирования пар строк путем сравнения содержимого соответствующих столбцов называется *объединением таблиц.* Таблица, получающаяся в результате формирования пар строк и содержащая данные из обеих таблиц, называется *объединением* двух таблиц. Объединение на основе точного равенства между двумя столбцами более правильно называется *объединением по равенству.* Объединения могут быть основаны на других видах сравнения столбцов (рассмотрены ниже).

Объединения представляют собой основу многотабличных запросов в SQL. В реляционной базе данных вся информация хранится в виде явных значений данных в столбцах, так что все возможные отношения между таблицами можно сформировать, сопоставляя содержимое соответствующих столбцов. Таким образом, объединения являются мощным (и, к тому же, единственным, ввиду отсутствия указателей) средством выявления отношений, существующих между данными.

Так как в SQL многотабличные запросы выполняются путем сопоставления столбцов, то инструкция *Select* для многотабличного запроса должна содержать условие отбора, которое определяет взаимосвязь между столбцами. Вот инструкция *Select* для запроса:

Вывести список ФИО, дату рождения и пол студентов, указав соответствующие специализации для каждого.

```
Select ФИО, [дата рождения], пол, [наименование специальности]
  From студенты, специальности
  Where студенты.[код специальности]= специальности.[код специальности]
```

```
ФИО                 дата рождения          пол        название специальности
Иванов А.И.       1983-12-12 00:00:00.000   мужской        ММ
Петрова И.И.      1982-11-01 00:00:00.000  женский         ПИ
Мухин М.А.        1982-05-14 00:00:00.000  мужской         СТ
Сидорова В.К.     1981-09-27 00:00:00.000  женский         МО
Пальчикова Н.Е.  1981-09-02 00:00:00.000  женский         ММ
Царегородцев Е.В. 1980-02-17 00:00:00.000  мужской         ПИ
Баранова Г.В.     1980-07-09 00:00:00.000  женский         СТ
Леухин П.Г.       1979-02-26 00:00:00.000  мужской         МО
Николаева А.П.    1979-03-17 00:00:00.000  женский         БУ
```

Приведенный запрос очень похож на простые запросы на выборку, однако между ними есть два отличия. Во-первых, предложение *from* содержит две таблицы, а не одну. Во-вторых, в условии отбора

```
студенты.[код специальности]= специальности.[код специальности]
```

сравниваются столбцы из двух различных таблиц. Эти столбцы называются *связанными.* Данное условие отбора, как и любое другое, уменьшает число строк в таблице результатов запроса. А поскольку запрос двухтабличный, условие отбора на самом деле уменьшает число *пар* строк в таблице результатов. В этом условии очень хорошо отражена суть сопоставления столбцов:

«Включить в таблицу результатов запроса только те пары строк, у которых *Код специальности* в таблице *Студенты* равен *Коду специальности* в таблице *Специальности*».

В инструкции *select* ничего не говорится о том, как должен выполняться запрос SQL. Там нет никаких упоминаний вроде "начните со студентов" или "начните со специальностей". Вместо этого в запросе сообщается, что должны представлять собой результаты запроса, а способ их получения оставлен на усмотрение СУБД.

Запросы с использованием отношения предок/потомок

Среди многотабличных запросов наиболее распространены запросы к двум таблицам, связанным с помощью отношения предок/потомок. Запрос о студентах и специальностях является примером такого запроса. у каждого студент (потомка) есть соответствующая ему специальность (предок), и на каждую специальность (предок) может быть много студентов (потомков). Пары строк, из которых формируются результаты запроса, связаны отношением предок/потомок.

В реляционной базе данных первичные и внешние ключи создают отношение предок/потомок. Таблица, содержащая внешний ключ, является потомком, а таблица с первичным ключом — предком. Чтобы использовать в запросе отношение предок/потомок, необходимо задать условие отбора, в котором первичный ключ сравнивается с внешним ключом. Вот пример такого запроса:

Вывести список студентов и предметов, по которым сдавался первый экзамен.

Таблица *Оценки* (потомок) содержит столбец *Код предмета*, который является внешним ключом для таблицы *Предметы* (предок). Здесь отношение предок/потомок используется с целью поиска в таблице *Предметы* для каждого студента соответствующей строки, содержащей город и регион, и включения ее в результат запроса.

В SQL не требуется, чтобы связанные столбцы были включены в результаты многотабличного запроса. На практике они чаще всего и не включаются, как это было в двух предыдущих примерах. Это связано с тем, что первичные и внешние ключи, как правило, представляют собой идентификаторы (такие как идентификатор специальности или идентификатор студента), которые человеку трудно запомнить, тогда как соответствующие названия (специальности, районы, имена, должности) запомнить гораздо легче. Поэтому вполне естественно, что в предложении *where для* объединения двух таблиц используются идентификаторы, а в предложении *select* для создания столбцов результатов запроса — более удобные для восприятия имена.

Условия для отбора строк

В многотабличном запросе можно комбинировать условие отбора, в котором задаются связанные столбцы, с другими условиями отбора, чтобы еще больше

сузить результаты запроса. Предположим, что требуется включить в результаты запроса только тех студентов, которые родились в Москве.

Вывести список ФИО, дату рождения и пол студентов, указав соответствующие специализации для каждого и адрес которых – Москва.

```
Select ФИО, [дата рождения], пол, [наименование специальности]
  From студенты, специальности
  Where студенты.[код специальности]= специальности.[код специальности]
   and адрес='Москва'
```

```
ФИО           дата рождения           пол  название специальности       адрес
Иванов А.И.  1983-12-12 00:00:00.000 мужской       ММ                  Москва
Петрова И.И. 1982-11-01 00:00:00.000 женский       ПИ                  Москва
```

Вследствие применения дополнительного условия отбора число строк в таблице результатов запроса уменьшилось. Согласно первому условию *студенты.[код специальности]= специальности.[код специальности*] из таблиц *студенты* и *специальности* отбираются пары строк, которые имеют соответствующее отношение предок/потомок; согласно второму условию производится дальнейший отбор только тех пар строк, где адрес равен Москве.

Несколько связанных столбцов

На практике бывает часто, что таблицы связаны парой составных ключей. Например, таблицы *orders* и *products* в учебной базе данных связаны парой составных ключей. Столбцы *mfr* и *product* в таблице *orders* вместе образуют внешний ключ для таблицы *products* и связаны с ее столбцами *mfr_id* и *product_id* соответственно. Чтобы объединить таблицы на основе такого отношения предок/потомок, необходимо задать *обе* пары связанных столбцов, как показано в данном примере:

Условие отбора в данном запросе показывает, что связанными парами строк таблиц *orders* и *products* являются те, в которых пары связанных столбцов содержат одни и те же значения Объединения посредством нескольких столбцов распространены меньше, чем объединения посредством одного столбца, и обычно встречаются в запросах с составными внешними ключами, как в приведенном выше примере.

Вывести список всех заказов, в том числе их стоимости и описания товаров

```
SELECT ORDER_NUM, AMOUNT, DESCRIPTION
FROM ORDERS, PRODUCTS
WHERE MFR = MFR_ID
AND PRODUCT = PRODUCT_ID
```

Условие отбора в данном запросе показывает, что связанными парами строк таблиц *orders* и *products* являются те, в которых пары связанных столбцов содержат одни и те же значения Объединения посредством нескольких столбцов распространены меньше, чем объединения посредством одного столбца, и обычно встречаются в запросах с составными внешними ключами.

Запросы на выборку к трем и более таблицам

SQL позволяет объединять данные из трех или более таблиц, используя ту же самую методику, что и для объединения данных из двух таблиц. В

промышленных приложениях нередко встречаются запросы к трем или четырем таблицам. Даже в рамках маленькой учебной базы данных, состоящей из четырех таблиц, нетрудно создать запрос к четырем таблицам, имеющий реальный смысл.

Вывести ФИО студентов, оценку по первому экзамену и наименование предмета, по которому проходил первый экзамен.

```
select ФИО, [оценка 1], [наименование предмета]
from студенты, оценки, предметы
where студенты.[код студента]= оценки.[код студента]
and оценки.[код предмета 1]=предметы.[код предмета]
```

ФИО	оценка 1	наименование предмета
Иванов А.И.	5	Операционные системы
Петрова И.И.	4	Проектирование информационных систем
Мухин М.А.	5	Базы данных
Сидорова В.К.	3	Офисные пакеты
Пальчикова Н.Е.	4	Языки програмирования
Царегородцев Е.В.	4	Офисные пакеты
Баранова Г.В.	2	Проектирование информационных систем
Леухин П.Г.	3	Базы данных
Николаева А.П.	5	Операционные системы
Петренко А.А	4	Языки програмирования

Прочие объединения таблиц по равенству

Огромное множество многотабличных запросов основано на отношениях предок/потомок, но в SQL не требуется, чтобы связанные столбцы представляли собой пару "внешний ключ — первичный ключ". Любые два столбца из двух таблиц могут быть связанными, если только они имеют сравнимые типы данных.

Объединение таблиц по неравенству

Термин "объединение" применяется к любому запросу, который объединяет данные из двух таблиц базы данных путем сравнения значений в двух столбцах этих таблиц. Самыми распространенными являются объединения, созданные на основе равенства связанных столбцов (объединения по равенству). Кроме того, SQL позволяет объединять таблицы с помощью других операций сравнения.

Особенности многотабличных запросов

Многотабличные запросы, рассмотренные до сих пор, не требовали применения специальных синтаксических форм или каких-либо других особенностей языка SQL помимо тех, что использовались для создания однотабличных запросов. Однако некоторые многотабличные запросы нельзя создать без использования дополнительных особенностей языка SQL, описанных в следующих параграфах. В частности:

- иногда в многотабличных запросах требуется использовать *полные имена столбцов,* чтобы исключить неоднозначные ссылки на столбцы;
- в многотабличных запросах особый смысл имеет *выбор всех столбцов* (*select* *);
- для создания многотабличных запросов, связывающих таблицу саму с собой, создаются *самообъединения;*
- в предложении *from* часто используются *псевдонимы таблиц,* чтобы упростить полные имена столбцов и обеспечить однозначность ссылок на столбцы в самообъединении.

Полные имена столбцов

В учебной базе данных имеется несколько случаев, когда две таблицы содержат столбцы с одинаковыми именами. Обычно с этими двумя столбцами

затруднений не возникает, поскольку в предложении *from* задается соответствующая таблица. Однако, если получать результаты многотабличного запроса из двух разных таблиц, может возникнуть неоднозначность. Чтобы исключить разночтения, при указании столбцов необходимо использовать их полные имена. Полное имя столбца содержит непосредственно имя столбца и имя таблицы, в которой он находится. Например, если в базе хранится текущий объем продаж *Sales* для таблицы *Office* и объем продаж каждого служащего *Sales* таблицы *Worker*, то полные имена двух столбцов *Sales* буду такими: *Office.Sales* и *Worker.Sales.*

В инструкции *select* вместо простых имен столбцов всегда можно использовать полные имена. Таблица, заданная в полном имени столбца, должна, конечно, соответствовать одной из таблиц, указанных в предложении *from***.**

Никогда не помешает использовать в многотабличных запросах полные имена столбцов. Недостатком, естественно, является то, что запросы становятся длиннее. В интерактивном режиме можно сначала попробовать выполнить запрос с простыми именами столбцов, чтобы найти все сомнительные столбцы. Если выдается сообщение об ошибке, запрос редактируют и точно указывают столбцы с повторяющимися именами.

Выборка всех столбцов

Инструкция *select* ***** используется для выборки всех столбцов таблицы, указанной в предложении *from.* В многотабличном запросе звездочка означает выбор всех столбцов из всех таблиц, указанных в предложении *from***.**

Сообщить всю информацию о студентах.

```
Select *
  From предметы,оценки
  Where предметы.[код предмета]=оценки.[код предметы 1]
```

Очевидно, что инструкция *select* ***** становится гораздо менее практичной, когда в предложении *from* указаны две, три или более таблицы.

Многие диалекты SQL трактуют звездочку как особый вид универсального имени столбца, которое преобразуется в список всех столбцов. В этих диалектах звездочка наряду с именем таблицы используется вместо списка полных имен столбцов. В следующем запросе имя *специальности* **. *** означает список имен всех столбцов таблицы *специальности.*

Псевдонимы таблиц

Псевдонимы таблиц необходимы в запросах, включающих самообъединения. Однако псевдоним можно использовать в любом запросе (например, если запрос касается таблицы другого пользователя или если имя таблицы очень длинное и употреблять его в полных именах столбцов утомительно). Следующий запрос ссылается на таблицу *специальности:*

Вывести список студентов с названиями специальности

```
Select ФИО,[название специальности]
  From студенты , специальности
  Where студенты.[код специальности]=специальности.[код специальности]
```

```
ФИО                                                название специальности
-------------------------------------------------- ----------------------
Иванов А.И.                                                        ММ
Петрова И.И.                                                       ПИ
```

```
Мухин М.А.                                    СТ
Сидорова В.К.                                 МО
Пальчикова Н.Е.                               ММ
Царегородцев Е.В.                             ПИ
Баранова Г.В.                                 СТ
Леухин П.Г.                                   МО
Николаева А.П.                                БУ
```

Если вместо имен двух таблиц использовать псевдонимы A и B, то и вводить, и читать этот запрос будет легче:

```
Select ФИО,[название специальности]
  From студенты A, специальности B
  Where A.[код специальности]=B.[код специальности]
```

Производительность при обработке многотабличных запросов

С увеличением количества таблиц в запросе резко возрастает объем работы, необходимой для выполнения запроса. В самом SQL нет ограничений на число таблиц, объединяемых в одном запросе. Но некоторые СУБД ограничивают число таблиц, чаще всего восемью. На практике высокие затраты на обработку многотабличных запросов во многих приложениях накладывают еще более сильные ограничения на количество таблиц.

В приложениях, предназначенных для оперативной обработки транзакций (OLTP), запрос обычно ссылается только на одну или две таблицы. В этих приложениях время ответа является критичной величиной: пользователь, как правило, вводит один или два элемента данных, и ему требуется получить ответ от базы данных в течение одной или двух секунд Вот некоторые типичные OLTP-запросы для учебной базы данных:

- пользователь вводит в какую-нибудь форму идентификатор клиента, и СУБД выводит на экран лимит кредита, состояние счета и другие данные об этом клиенте (запрос к одной таблице);
- пользователь с помощью специального устройства сканирует с упаковки номер товара, и СУБД выводит на экран наименование и цену товара (запрос к одной таблице);
- пользователь вводит имя служащего, и программа выдает список текущих заказов, принятых данным служащим (запрос к двум таблицам).

В отличие от OLTP-приложений, в приложениях, предназначенных для поддержки принятия решений, запрос, как правило, обращается ко многим таблицам и использует сложные отношения, существующие в базе данных В этих приложениях результаты запроса часто нужны для принятия важных решений, поэтому вполне приемлемыми считаются запросы, которые выполняются несколько минут или даже несколько часов. Вот типичные запросы, связанные с принятием решений:

- пользователь вводит название офиса, и программа выдает список двадцати пяти самых больших заказов, принятых служащими этого офиса (запрос к трем таблицам);
- в отчете суммируются продажи каждого служащего по типам товаров и показывается, какой служащий какие товары продал (запрос к трем таблицам);
- руководитель рассматривает возможность открытия нового офиса в Сиэтле и выполняет запрос для анализа заказов, клиентов, товаров и служащих (запрос к четырем таблицам).

Внутренняя структура объединения таблиц

Для простых объединений таблиц довольно легко написать правильную инструкцию *select* на основе запроса, выраженного на обычном языке. И наоборот, глядя на инструкцию *select*, можно легко определить, что она делает. Однако если в *объединении* участвует много таблиц или используется сложное условие отбора, бывает не так-то просто понять, какую функцию выполняет та или иная инструкция *select.* По этой причине необходимо дать более точное определение понятию "объединение".

Умножение таблиц

Объединение — это частный случай более общей комбинации данных из двух таблиц, известной под названием *декартово произведение* (или просто *произведение)* двух таблиц. Произведение двух таблиц представляет собой таблицу (называемую *таблицей произведения),* состоящую из всех возможных пар строк обеих таблиц Столбцами таблицы произведения являются все столбцы первой таблицы, за которыми следуют все столбцы второй таблицы. На рис. 1 изображен пример произведения двух маленьких таблиц.

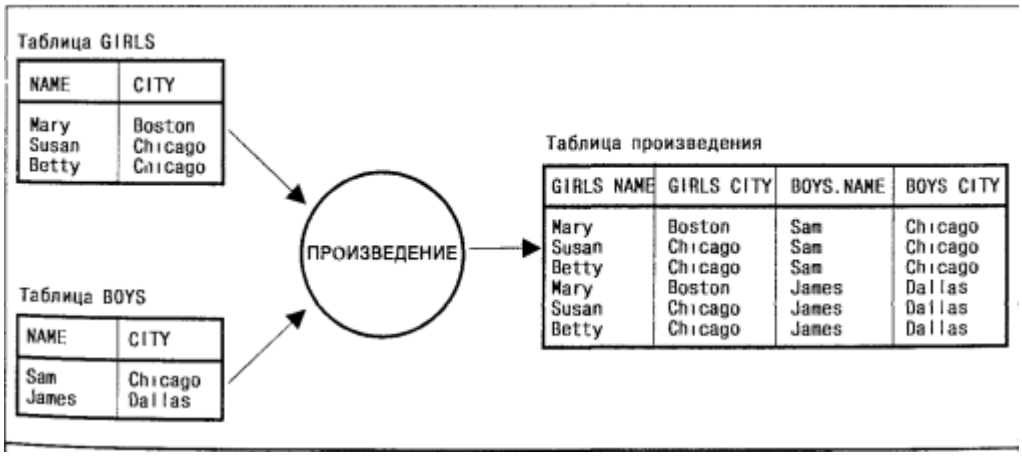

Рис. 1 Произведение двух таблиц

Если создать запрос к двум таблицам без предложения ***where***, то таблица результатов запроса окажется произведением двух таблиц.

Показать всевозможные комбинации студентов и специальностей

```
Select ФИО,[название специальности]
  From студенты A, специальности B
```

Для объединения двух упомянутых таблиц используется точно такая же инструкция ***select***, но только с предложением ***where***, содержащим условие сравнения связанных столбцов:

```
Select ФИО,[название специальности]
  From студенты , специальности
  Where студенты.[код специальности]=специальности.[код специальности]
```

Два приведенных выше запроса указывают на важную связь между объединениями и произведением:

"Объединение двух таблиц является произведением этих таблиц, из которого удалены некоторые строки. Удаляются именно те строки, которые не удовлетворяют условию сравнения связанных столбцов (условию отбора) для данного объединения".

Правила выполнения многотабличных запросов на выборку

В табл. 1 представлены правила выполнения SQL-запроса на выборку, расширенные для случая многотабличных запросов. Эти правила раскрывают смысл любой многотабличной инструкции ***select, в*** точности определяя процедуру, которая всегда позволяет получить правильный набор результатов запроса.

Таблица результатов запроса на выборку генерируется следующим образом:
1. Если запрос представляет собой запрос на объединение (UNION) инструкций SELECT, для каждой из этих инструкций выполнить действия 2—5 и получить отдельную таблицу результатов.
2. Сформировать произведение таблиц, перечисленных в предложении FROM. Если в предложении FROM указана только одна таблица, то произведением будет она сама
3. Если имеется предложение WHERE, применить заданное в нем условие отбора к каждой строке таблицы произведения и оставить в ней только те строки, для которых это условие выполняется, т.е. имеет значение TRUE; строки, для которых условие отбора имеет значение FALSE или NULL, — отбросить.
4 Для каждой из оставшихся строк вычислить значение каждого элемента в списке возвращаемых столбцов и создать одну строку таблицы результатов запроса При любой ссылке на столбец берется значение столбца для текущей строки.
5. Если указан предикат DISTINCT, удалить из таблицы результатов запроса все повторяющиеся строки.
6. Если запрос является запросом на объединение (UNION) инструкций SELECT, объединить результаты выполнения отдельных инструкций в одну таблицу результатов запроса. Удалить из нее повторяющиеся строки, если не указан предикат ALL.
7. Если имеется предложение ORDER BY, отсортировать результаты запроса.

Таблица 1. Правила выполнения многотабличного SQL-запроса на выборку

Объединения и стандарт SQL2

Для создателей стандарта SQL2 внешние объединения были серьезной проблемой. Так как внешние объединения — это единственный способ представления результатов ряда крайне необходимых запросов, было важно, чтобы стандарт SQL2 поддерживал данное понятие. Кроме того, внешние объединения поддерживались многими СУБД и становились все более важной частью SQL. Однако, как видно из предыдущих параграфов, способы представления внешних объединений в различных СУБД сильно отличались друг от друга. Кроме того, все способы обозначения внешних объединений в коммерческих программных продуктах страдали недостатками и выбирались по принципу их минимального влияния на язык SQL, а не из соображений ясности и точности.

На этом фоне в стандарте SQL2 был определен совершенно новый метод поддержки внешних объединений, который не опирался ни на одну популярную СУБД. В спецификации стандарта SQL2 поддержка внешних объединений осуществляется в предложении ***from*** с тщательно разработанным синтаксисом, позволяющим пользователю точно определить, каким образом исходные таблицы должны быть объединены в запросе. Механизм поддержки внешних объединений, представленный в стандарте SQL2, обладает двумя преимуществами. Во-первых, с появлением стандарта SQL2 стало возможным создавать объединения самых сложных видов. Во-вторых, существующие СУБД могут без каких-либо конфликтов поддерживать имеющиеся в стандарте SQL2 расширения стандарта SQL1 и в то же время сохранять поддержку своего собственного синтаксиса для внешних объединений.

Этих преимуществ удалось добиться за счет значительного усложнения прежде одного из самых простых разделов языка SQL. По сути, расширенная поддержка объединений является частью значительного расширения возможностей

запросов в SQL2 в целом. Стало возможным выполнять операции над результатами запроса как над множествами (сложение, пересечение, вычитание таблиц) и применять выражения, манипулирующие строками, таблицами и подчиненными запросами.

Внутренние объединения в стандарте SQL2

На рис. 2 в упрощенном виде изображена синтаксическая диаграмма предложения ***from в*** стандарте SQL2. Изучить все ее варианты легче всего, рассматривая по очереди каждый тип объединения.

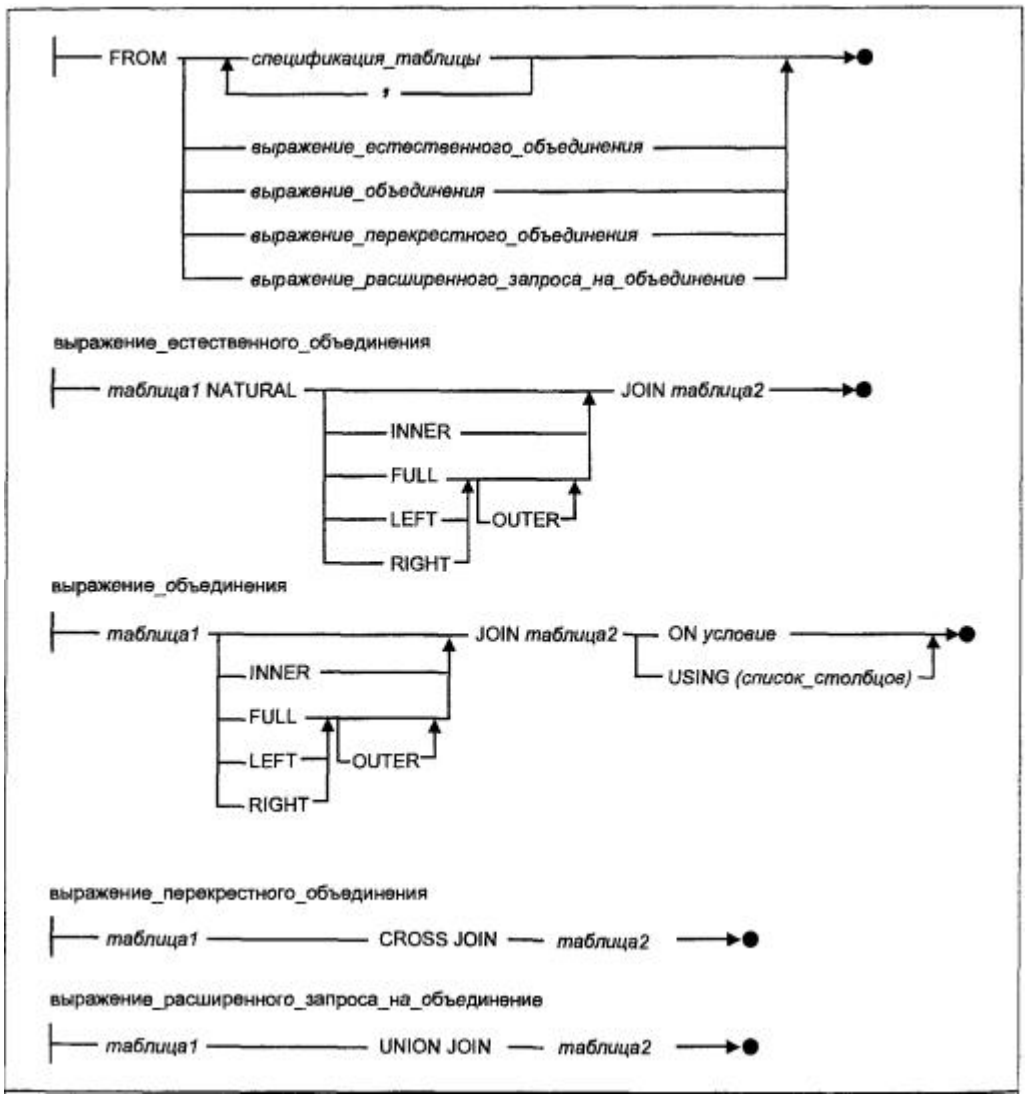

Рис. 2 Расширенное предложение From в стандарте SQL2

Например, стандартное внутреннее объединение таблиц ***girls*** и ***boys,*** представленных на рисунке 3,

Таблица GIRLS

NAME	CITY
Mary	Boston
Nancy	NULL
Susan	Chicago
Betty	Chicago
Anne	Denver

Таблица BOYS

NAME	CITY
John	Boston
Henry	Boston
George	NULL
Sam	Chicago
James	Dallas

Рис. 3 Стурктура таблиц Bots и Girls

на языке SQL1 можно выразить так

```
SELECT *
  FROM GIRLS, BOYS
  WHERE GIRLS.CITY = BOYS.CITY
```

В стандарте SQL2 это по-прежнему допустимая инструкция. Создатели стандарта SQL2 не могли отказаться от подобного синтаксиса, так как он применяется в миллионах работающих приложений. Тем не менее, инструкцию можно также записать следующим образом:

```
SELECT *
  FROM GIRLS INNER JOIN BOYS ON GIRLS.CITY = BOYS.CITY
```

Две таблицы объединяются явно *посредством* операции *join*, а условие отбора, описывающее объединение, находится теперь *в* предложении *on* внутри предложения *from.* В условии отбора могут быть заданы любые критерии сравнения строк двух объединяемых таблиц. Предположим, например, что таблицы *boys* и *girls* были расширены путем добавления столбца *age. Вот* объединение, в котором связываются пары девочка/мальчик из одного города, а также требуется, чтобы мальчик и девочка в каждой паре были одного возраста:

```
select *
  from girls inner join boys
  on (girls.city = boys.city)
  and (girls.age = boys.age)
```

В этих простых двухтабличных объединениях все содержимое предложения *where* просто перешло в предложение *on,* т.е. последнее не добавляет ничего нового в язык SQL. Вспомните, однако, что во внешних объединениях трех и более таблиц результат запроса зависит от порядка, в котором производятся объединения. Как описывается ниже в настоящей главе, с помощью предложения *on* осуществляется управление процессом обработки таких многотабличных объединений.

Стандарт SQL2 допускает еще один вариант запроса на простое внутреннее объединение таблиц *girls и boys*. Так как связанные столбцы этих таблиц имеют одинаковые имена и сравниваются на предмет равенства (что делается довольно часто), то можно использовать альтернативную форму предложения *on,* в которой задается список имен связанных столбцов:

```
select *
  from girls inner join boys
  using (city, age)
```

В предложении *using* перечисляются через запятую имена связанных столбцов; они должны быть идентичными в обеих таблицах. Это предложение полностью эквивалентно предложению *on,* в котором каждая пара связанных столбцов задается явно, но намного компактнее и, следовательно, легче для понимания. Конечно, если связанные столбцы имеют разные имена в таблицах *boys и girls, то* необходимо использовать предложение *on* или *where* со знаком равенства. Предложение *on* требуется использовать также в том случае, если объединение не производится по равенству связанных столбцов. Например, если вы хотите выбрать пары девочка/мальчик, в которых девочка старше мальчика, то должны задать объединение с помощью предложения *on:*

```
select *
  from girls inner join boys
  on (girls.city = boys.city and girls.age > boys.age)
```

Наконец, имеется еще один, последний вариант этого простого запроса, иллюстрирующий одну особенность предложения *from в* стандарте SQL2. Объединение двух таблиц, в котором связанные столбцы имеют идентичные имена, называется *естественным объединением,* так как обычно это действительно самый "естественный" способ объединения двух таблиц. Запрос на выборку пар девочка/мальчик, живущих в одних и тех же городах и имеющих тот же самый возраст, можно выразить как естественное объединение следующим образом:

```
select *
from girls natural inner join boys
```

Если задано ключевое слово *natural*, предложения *on* и *using* могут отсутствовать в запросе на объединение, так как в естественном объединении точно определено условие отбора для объединения двух таблиц — сравниваются все столбцы с идентичными именами в обеих таблицах.

В стандарте SQL2 определено, что объединение двух таблиц по умолчанию является внутренним. Во всех предыдущих примерах можно опустить ключевое слово *inner, и* полученные в результате запросы по-прежнему будут действительными инструкциями стандарта SQL2, имеющими тот же самый смысл.

Внешние объединения в стандарте SQL2 *

Стандарт SQL2 обеспечивает полную поддержку внешних объединений, расширяя языковые конструкции, используемые для внутренних объединений. Например, полное внешнее объединение таблиц *girls* и *boys* (без столбцов *age*) создается следующим запросом:

```
SELECT *
FROM GIRLS FULL OUTER JOIN BOYS ON GIRLS.CITY = BOYS.CITY
```

Как объяснялось ранее в настоящей главе, таблица результатов запроса будет содержать по одной строке для каждой связанной пары девочка/мальчик, а также по одной строке для каждой несвязанной записи для девочки или мальчика, расширенной значениями *null в* столбцах другой таблицы. В стандарте SQL2 для внешних объединений допустимы те же самые вариации, что и для внутренних объединений. Данный запрос можно было бы записать так:

```
SELECT *
FROM GIRLS NATURAL FULL OUTER JOIN BOYS
```

Или так:

```
SELECT *
FROM GIRLS FULL OUTER JOIN BOYS USING (CITY)
```

Ключевое слово *outer*, так же как и ключевое слово *inner,* в стандарте SQL2 не является обязательным. Поэтому предыдущий запрос можно было бы переписать следующим образом:

```
SELECT *
FROM GIRLS FULL JOIN BOYS USING (CITY)
```

По слову *full СУБД* сама определяет, что запрашивается внешнее *объединение.* Вполне естественно, что в стандарте *SQL2* левое и правое внешние *объединения* обозначаются словами *left и right* вместо слова *full*. Вот вариант того же запроса, определяющий левое внешнее объединение:

```
SELECT *
```

```
FROM GIRLS LEFT OUTER JOIN BOYS USING (CITY)
```

Как уже говорилось, в результаты запроса войдут строки связанных пар девочка/мальчик и все несвязанные строки, содержащие *null,* из таблицы *girls, но* не *войдут* несвязанные строки из таблицы *boys*. И наоборот, вот вариант этого же *запроса,* задающий правое внешнее объединение:

```
select *
from girls right outer join boys using (city)
```

В данном случае в таблице результатов будут представлены пары девочка/мальчик и несвязанные строки из таблицы *boys* ("правая" таблица объединения), но будут отсутствовать несвязанные строки из таблицы *girls.*

Перекрестные объединения и расширенные запросы на объединение в SQL2

Расширенное предложение *from* в стандарте SQL2 поддерживает также два других способа объединения данных из двух таблиц — декартово произведение (описанное ранее в настоящей главе) и расширенный запрос на объединение. Вот запрос, создающий произведение таблиц *girls* и *boys:*

```
select *
from girls cross join boys
```

По определению, декартово произведение (иногда называемое *перекрестным объединением* (cross join), отсюда и название операции ***cross join***) содержит все возможные пары строк из двух таблиц. Оно является результатом "умножения" двух таблиц, превращая таблицы трех девочек и двух мальчиков в таблицу шести пар девочка/мальчик (3*2=6). Перекрестным объединениям не сопутствуют никакие "связанные столбцы" или "условия отбора", поэтому предложения *on* и *using в них* не допускаются. Следует отметить, что операция перекрестного объединения не добавляет ничего нового к возможностям SQL. Те же результаты можно получить с помощью внутреннего объединения, если не задать в нем условия отбора. Поэтому предыдущий запрос можно переписать так:

```
select *
from girls, boys
```

Ключевые слова *cross join* в предложении *from* просто в явном виде указывают на то, что создается декартово произведение. Для большинства баз данных такая операция вряд ли будет представлять практический интерес. Она полезна лишь тогда, когда на основе полученной таблицы строятся более сложные выражения, например итоговые запросы (рассматриваются в следующей главе).

Расширенный запрос на объединение заключает в себе некоторые черты как операции *union* (рассматривалась в предыдущей главе), так и операции *join.* Вспомним, что операция *union* объединяет строки двух таблиц, у которых должно быть одинаковое число столбцов, причем типы соответствующих столбцов обеих таблиц должны быть совместимыми. Простейший запрос на объединение

```
select *
from girls
union all
SELECT *
FROM BOYS
```

будучи примененным к таблице *girls*, состоящей из трех строк, и таблице *boys*, содержащей две строки, возвратит таблицу из пяти строк. Каждая из них в точности соответствует одной из строк таблицы *girls* либо *boys*. В таблице результатов запроса будет два столбца, *name и city*, как и в обеих исходных таблицах.

Расширенный запрос на объединение к этим же таблицам записывается так:

```
SELECT *
FROM GIRLS UNION JOIN BOYS
```

Таблица результатов такого запроса снова будет содержать пять строк, каждая из которых образована в точности одной строкой из таблиц *girls* либо *boys*. Но, в отличие от предыдущего случая, количество столбцов в полученной таблице будет четыре, а не два, — все столбцы из первой таблицы *плюс* все столбцы из второй таблицы. Здесь можно найти определенное сходство с внешним объединением. В каждой строке таблицы результатов запроса, построенной на основе строки из таблицы *girls*, значения в столбцах, взятых из таблицы *girls*, будут оставлены без изменений; все остальные столбцы (из таблицы *boys*) заполняются значениями *null*. *Точно* так же в каждой строке из таблицы *boys* значения в столбцах этой таблицы остаются неизменными, а столбцы из таблицы *girls* заполняются значениями *null*.

В заключение полезно будет провести сравнение различий между наборами результатов, получаемыми при выполнении объединений различных видов. При объединении двух таблиц, *tbl1 с* числом строк *m* и *tbl2* с числом строк *n*, происходит следующее:

- *перекрестное объединение* вернет таблицу с числом строк *mx n*, состоящую из всех возможных пар строк обеих таблиц;
- *внутреннее объединение* вернет таблицу, состоящую из некоторого числа строк, *r*, причем $r < m*n$. Внутреннее объединение является подмножеством декартового произведения. Оно образуется путем удаления тех строк из таблицы произведения, которые не удовлетворяют условию отбора;
- *левое внешнее объединение* вернет таблицу, содержащую все строки внутреннего объединения плюс расширенные значениями *null* строки таблицы *tbl1*, не удовлетворяющие условию отбора;
- *правое внешнее объединение* вернет таблицу, содержащую все строки *внутреннего* объединения плюс расширенные значениями *null* строки таблицы *tbl2*, не удовлетворяющие условию отбора;
- *полное внешнее объединение* вернет таблицу, содержащую все строки *внутреннего* объединения плюс расширенные значениями *null* строки таблиц *tbl1 и tbl2*, не удовлетворяющие условию отбора;
- *расширенный запрос на объединение* вернет таблицу, содержащую все строки таблиц *tbl1* и *tbl2*, расширенные значениями *null*.

Многотабличные объединения в стандарте SQL2

Одно из важных преимуществ расширенного предложения ***from*** заключается в том, что оно дает единый стандарт для определения всех возможных видов объединений. Другим, даже еще более значительным преимуществом этого

предложения является то, что оно обеспечивает четкую спецификацию объединений трех и более таблиц. При построении столь сложных объединений любое выражение объединения может быть заключено в круглые скобки. Результирующее выражение, в свою очередь, можно использовать для создания других выражений объединения, как если бы оно было простой таблицей. Точно так же, как SQL позволяет с помощью круглых скобок комбинировать различные арифметические операции (+,-,* и /), стандарт SQL2 дает возможность создавать сложные выражения для объединений.

Чтобы привести пример многотабличного объединения, предположим, что к таблицам *girls* и *boys* добавлена новая таблица *parents*, которая имеет три столбца:

Child Соответствует столбцу *name* в таблицах *girls* и *boys*
Type Принимает значение *father* (отец) или *mother* (мать)
Pname Имя родителя

Строка в таблице *girls* или *boys* может иметь две связанные строки в таблице *Parent*, одна из которых определяет мать, а другая — отца, или может иметь только одну из этих строк, или может совсем не иметь связанных строк, если отсутствуют данные о родителях ребенка. В таблицах *girls, boys и parents* в совокупности содержится достаточно богатый набор данных, чтобы обеспечить несколько примеров многотабличных запросов.

Предположим, например, что вы хотите составить список всех девочек и их матерей, а также мальчиков, которые живут в том же городе. Вот запрос, создающий такой список:

```
SELECT GIRLS.NAME, PNAME, BOYS.NAME
FROM (GIRLS INNER JOIN PARENTS
  ON PARENTS.CHILD = GIRLS.NAME)
INNER JOIN BOYS
  ON GIRLS.CITY = BOYS.CITY
WHERE TYPE = "MOTHER"
```

Так как оба объединения являются внутренними, то девочки, не имеющие связанных строк в таблицах *parents и boys, в* таблицу результатов запроса не попадут. Модифицировав первую часть запроса в левое внешнее объединение, можно включить в результаты запроса строки, соответствующие девочкам, для которых отсутствует информация о матерях:

```
SELECT GIRLS.NAME, PNAME, BOYS.NAME
FROM (GIRLS LEFT JOIN PARENTS
 ON PARENTS.CHILD = GIRLS.NAME)
INNER JOIN BOYS
 ON GIRLS.CITY = BOYS.CITY
WHERE (TYPE = "MOTHER") OR (TYPE IS NULL)
```

В результатах этого запроса по-прежнему отсутствуют девочки, в одном городе с которыми не живет ни одного мальчика. Если их тоже нужно включить в запрос, следует и второе объединение сделать внешним:

```
SELECT      GIRLS.NAME, PNAME, BOYS.NAME
FROM (GIRLS LEFT JOIN PARENTS
 ON PARENTS.CHILD = GIRLS.NAME)
LEFT  JOIN BOYS
 ON   GIRLS.CITY = BOYS.CITY
WHERE (TYPE = "MOTHER") OR (TYPE IS NULL)
```

Следует обратить внимание на то, что левое внешнее объединение таблиц *girls* и *parents* создает одну дополнительную проблему. Если для какой-либо девочки отсутствует информация о матери, то не только столбец *pname,* но и столбец *type* будет равен *null*. Поэтому простая проверка

```
WHERE (TYPE = "MOTHER")
```

для таких строк вернет значение *null, и* они не будут включены в результаты запроса. Но ведь это противоречит смыслу самого запроса! Для решения данной проблемы в предложение *where* добавлена вторая проверка на равенство столбца *type* значению *null.*

В качестве завершающего примера предположим, что вы опять, как в предыдущих случаях, хотите найти пары девочка/мальчик из одного и того же города, но на этот раз вы хотите еще и включить в таблицу результатов имя отца мальчика и имя матери девочки. Такой запрос требует четырехтабличного объединения (таблицы *boys, girls* и две копии таблицы *parents*: одна для объединения с таблицей *boys с* целью *получения* имен отцов, а вторая для объединения с таблицей *girls с* целью *получения* имен матерей). *То,* что в объединениях данного примера могут присутствовать несвязанные строки, означает существование нескольких возможных "пра*вильных"* ответов на запрос. Например, допустим, что вам нужно включить в результаты запроса все пары девочка/мальчик из одних *и* тех городов, даже те, в которых либо мальчик, либо девочка не имеют связанных строк с таблицей *parents*. В этом *запросе* придется использовать два внешних объединения — между таблицами *boys* и *parents*, а также таблицами *girls* и *parents, — и* одно внутреннее: между таблицами *boys и girls* Согласно стандарту SQL2, этот запрос будет выглядеть следующим образом:

```
SELECT GIRLS.NAME, MOTHERS.PNAME, BOYS.NAME, FATHERS.PNAME
FROM (GIRLS LEFT JOIN PARENTS AS MOTHERS
 ON ((MOTHERS.CHILD = GIRLS.NAME) AND (MOTHERS.TYPE = "MOTHER"))
INNER JOIN (BOYS LEFT JOIN PARENTS AS FATHERS
 ON ((FATHERS.CHILD = BOYS.NAME) AND (FATHERS.TYPE = "FATHER")) USING
(CITY)
```

В этом запросе проблема проверки столбца type в предложении where решена по-другому: сама проверка перемещена в предложение On обеих операций объединения. В этом случае столбец type будет проверяться на этапе построения каждого объединения, когда строки, расширенные значениями null, еще не добавлены в таблицу результатов запроса. Поскольку таблица parents встречается в предложениях from дважды в разных ролях, необходимо назначить ей два псевдонима, чтобы в предложении select можно было указать правильные столбцы.

Как видно из примера, даже запрос с тремя объединениями в соответствии со стандартом SQL2 может иметь весьма сложный вид. Однако, несмотря на эту сложность, запрос по стандарту SQL2 точно и однозначно определяет то, что должна выполнить СУБД. Нет никакой неясности в отношении порядка объединения таблиц или в отношении того, какие объединения являются внешними, а какие — внутренними. В общем, новые возможности стоят дополнительных сложностей расширенного предложения from стандарта SQL2.

Предложения *where и order by* могут свободно использоваться с расширенным предложением *from*. Сначала выполняются операции, указанные в предложении *from,* включая все объединения или запросы на объединение. Условия объединения, заданные в предложении *using* или *on*, выступают как часть

конкретного объединения, по отношению к которому они определены. Когда выполнение операций в предложении *from* заканчивается, к таблице результатов применяются условия отбора, заданные в предложении *where*. Таким образом, в предложении *on* задаются Условия отбора, применяемые к отдельным объединениям; в предложении *where* задается условие отбора, которое применяется к результирующей таблице этих объединений.

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. Поясните смысл понятия **объединение таблиц**.
2. Поясните смысл понятия **объединение по равенству**.
3. Дайте определение понятию **связные столбцы**.
4. Дайте определение понятию **объединение**.
5. Дайте определение понятию **естественное соединение**.

Преподаватель С.В. Банцевич

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