МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное автономное образовательное учреждение
высшего профессионального образования
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

ЧЕРДЫНЦЕВА М.И.

ТЕХНОЛОГИИ БАЗ ДАННЫХ. ЯЗЫК SQL

МЕТОДИЧЕСКИЕ УКАЗАНИЯ
для студентов 3-4 курсов дневного и вечернего отделений
факультета математики, механики и компьютерных наук

Ростов-на-Дону
2010

Методические указания разработаны сотрудником кафедры прикладной математики и программирования: кандидатом технических наук, доцентом М.И. Чердынцевой.

Компьютерный набор и верстка доцент Чердынцева М.И.

Печатается в соответствии с решением кафедры прикладной математики и программирования факультета математики, механики и компьютерных наук ЮФУ, протокол № 10 от 17 июня 2010 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Рост популярности языка SQL – одна из самых важных тенденций современной компьютерной промышленности. За последние десятилетия SQL стал стандартным языком баз данных. Язык SQL является важным звеном в архитектуре систем управления базами данных, выпускаемых всеми ведущими поставщиками программных продуктов.

Нужно заметить, что в настоящее время, ни одна система не реализует стандарт SQL в полном объеме. Кроме того, во всех диалектах языка имеются возможности, не являющиеся стандартными. Таким образом, можно сказать, что каждый диалект – это надмножество некоторого подмножества стандарта SQL. Тем не менее, изучив конкретный диалект одной из СУБД, достаточно просто в дальнейшем знакомиться с версиями языка других фирм – разработчиков.

В методических указаниях представлены сведения об основных средствах языка SQL в части подъязыка манипулирования данными (DML – Data Manipulation Language). Представленный в методических указаниях материал используется при изучении модуля «Языки реляционных бах данных. Язык SQL» курсов «Технологии баз данных» и «Базы данных и экспертные системы». Основную часть методических указаний составляют примеры решения задач. Методические указания могут быть использованы для проведения лабораторных работ по курсу «Технологии баз данных» и студентами при выполнении самостоятельной работы и решении индивидуальных заданий.

Целью данных методических указаний является демонстрация возможностей языка SQL на конкретных примерах. Поскольку методические указания предназначены для проведения лабораторных работ студентов, изложение ведется применительно к версии языка SQL, реализованной в СУБД Firebird.

1 ЯЗЫК РЕЛЯЦИОННЫХ БАЗ ДАННЫХ SQL

Структурированный язык запросов SQL (Structured Query Language) является языком, предназначенным для обработки и извлечения данных, содержащихся в базе данных.

Язык SQL применяется для организации взаимодействия пользователя с *реляционной* базой данных.

Язык SQL является реляционно-полным. Это означает, что любой оператор реляционной алгебры может быть выражен подходящим оператором SQL.

Язык SQL оперирует терминами, несколько отличающимися от терминов реляционной теории. Например, вместо термина "отношения" используется – "таблицы". Вместо "кортежей" – "строки", вместо "атрибутов" – "колонки" или "столбцы". Стандарт языка SQL, хотя и основан на реляционной теории, но во многих местах отходит он нее. Например, отношение в реляционной модели данных не допускает наличия одинаковых кортежей, а таблицы в терминологии SQL могут иметь одинаковые строки.

В настоящий момент язык SQL представляет собой нечто гораздо большее, чем простой инструмент создания запросов, хотя первоначально он именно для этого был и предназначен. Сейчас этот язык используется для реализации всех функциональных возможностей, которые СУБД предоставляет пользователю, а именно:

1. SQL дает возможность изменять структуру представления данных, а также устанавливать отношения между элементами базы данных, т.е. реализует функцию *организации* данных.
2. SQL позволяет *манипулировать* данными, т.е. читать данные из базы данных, добавлять новые данные, удалять или изменять уже имеющиеся в базе данные. При этом реализуются функции *чтения и обработки* данных.

3. С помощью SQL можно защитить данные от несанкционированного доступа. Это обеспечивается средствами *авторизации доступа*.
4. SQL позволяет координировать совместное использование данных многими пользователями, работающими параллельно, чтобы они не мешали друг другу. Это функция *совместного использования данных.*
5. SQL позволяет обеспечить *целостность базы данных*, защищая ее от разрушения из-за ошибок пользователей, несогласованных изменений или отказа системы.

Следующие разделы методических указаний посвящены описанию и примерам использования группы операторов языка SQL реализующих функцию *манипулирования* данными. Это подмножество операторов называется DML (Data Manipulation Language) – операторы манипулирования данными. В DML входят следующие операторы:

- `SELECT` – запросить данные из таблиц
- `INSERT` - добавить строки в таблицу
- `UPDATE` - изменить строки в таблице
- `DELETE` - удалить строки в таблице

В методических указаниях рассмотрен синтаксис DML , поддерживаемый СУБД Firebird. За более подробной информацией о правилах написания запросов следует обратиться к документации по конкретной СУБД.

2 ОПЕРАТОР `SELECT`

Оператор `SELECT` является фактически основным и самым сложным оператором SQL. Он предназначен для выборки данных из таблиц, именно он и реализует одно их основных назначение базы данных – предоставлять по запросу информацию из базы данных пользователю. Поскольку язык SQL является

не процедурным, а декларативным, оператор SELECT предназначен для того, чтобы описать какие данные должны быть получены из базы данных в результате выполнения запроса.

Полное описание синтаксиса оператора SELECT в СУБД Firebird (как и в стандарте SQL) является достаточно сложным. С ним можно ознакомиться в документации или в литературе по Firebird.

Для начального ознакомления рассмотрим упрощенную форму оператора:

```
SELECT
[FIRST (m)] [SKIP (n)][DISTINCT|ALL]
<список столбцов>|*
     FROM <таблица> [<алиас>]|<процедура> [<алиас>]|
     <просмотр> [<алиас>]|<соединение таблиц>
[WHERE <условия отбора строк>]
[GROUP BY <список столбцов для группировки>
  [HAVING <условия отбора групп>]]
[UNION <оператор select> [ALL]]
[ORDER BY <список сортировки>]
```

Оператор SELECT всегда выполняется над некоторыми таблицами, входящими в базу данных.

На самом деле в базах данных могут быть не только постоянно хранимые таблицы, а также временные таблицы и, так называемые, представления. Представления – это просто хранящиеся в базе данных SELECT–выражения. С точки зрения пользователей представления – это таблица, которая не хранится постоянно в базе данных, а "возникает" в момент обращения к ней. С точки зрения оператора SELECT и постоянно хранимые таблицы, и временные таблицы и представления выглядят совершенно одинаково. Конечно, при реальном выполнении оператора SELECT системой учитываются различия между хранимыми таблицами и представлениями, но эти различия скрыты от пользователя.

Результатом выполнения оператора SELECT всегда является таблица. Таким образом, по результатам действий оператор SELECT похож на операторы реляционной алгебры. Любой оператор реляционной алгебры может быть выражен оператором SELECT. Сложность оператора SELECT определяется тем, что он содержит в себе все возможности реляционной алгебры, а также дополнительные возможности, которых в реляционной алгебре нет.

2.1 Запросы с использованием данных одной таблицы

Приводимые ниже примеры показывают, как получить различную информацию из одной таблицы базы данных.

Непосредственно за предложением SELECT следует описание столбцов, которые будут включены в результирующую таблицу. Это могут быть имена столбцов той таблицы (таблиц), для которой выполняется запрос, выражения, константы. Если необходимо включить в результат все столбцы из исходной таблицы, используется сокращенное обозначение *.

Предложение FROM предназначено для спецификации списка таблиц, содержащих данные, которые считывает запрос. Для однотабличных запросов список всегда содержит имя одной таблицы (или представления, или процедуры).

Предложение WHERE показывает, что в результат запроса следует включить только некоторые строки. Для отбора строк, включаемых в результат запроса, используется условие поиска. В качестве условия поиска можно использовать сложные логические выражения, включающие имена полей таблиц, константы, операции сравнения (>, <, = , IS NULL и т.д.), скобки, логические операции AND ,OR и NOT. Более полное описание приведено в разделе «Выражения и предикаты».

Пример 1. Выбрать все данные из таблицы сотрудников.

```
SELECT *
     FROM EMPLOYEE;
```

Результатом выполнения данного запроса будет таблица – точная копия таблицы EMPLOYEE. Вместо перечисления всех столбцов таблицы EMPLOYEE указано *.

Пример 2. Выбрать все сведения о сотрудниках, работающих в отделе номер 120.

```
SELECT *
     FROM EMPLOYEE
     WHERE DEPT_NO = 120;
```

Пример 3. Выдать все сведения об отделах, в которых нет главного менеджера.

```
SELECT *
     FROM DEPARTMENT
     WHERE MNGR_NO IS NULL;
```

Пример 4. Получить список всех сотрудников, с указанием полного имени, оклада и даты приема на работу.

```
SELECT FULL_NAME, SALARY, HIRE_DATE
     FROM EMPLOYEE;
```

Пример 5. Получить список стран, в которых работают сотрудники фирмы.

```
SELECT JOB_COUNTRY
          FROM EMPLOYEE;
```

Результатом данного запроса будет таблица, содержащая один столбец и столько строк, сколько всего сотрудников имеется в базе таблице EMPLOYEE. Поскольку многие из них работают в одной стране, будет выдано много повторяющихся строк.

Пример 6. Получить список стран, в которых работают сотрудники фирмы, без повторений.

```
SELECT DISTINCT JOB_COUNTRY
          FROM EMPLOYEE;
```

Для исключения повторяющихся строк используется DISTINCT.

Пример 7. Выдать список адресов всех фирм – клиентов. Адрес должен включать сведения из колонок ADDRESS_LINE1, CITY, COUNTRY. Чтобы адрес представлял единое целое, сформируем в запросе новую колонку

```
SELECT CUSTOMER, ADDRESS_LINE1||CITY||COUNTRY AS ADDRESS
          FROM CUSTOMER;
```

Пример 8. Выдать список сотрудников, упорядочив его по возрастанию величины оклада.

```
SELECT FULL_NAME
          FROM EMPLOYEE
          ORDER BY SALARY;
```

Пример 9. Выдать список сотрудников, упорядочив его по убыванию величины оклада.

```
SELECT FULL_NAME
          FROM EMPLOYEE
     ORDER BY SALARY DESC;
```

Пример 10. Выдать список из пяти самых высокооплачиваемых сотрудников.

```
SELECT FIRST (5) FULL_NAME
          FROM EMPLOYEE
     ORDER BY SALARY DESC;
```

Пример 11. Определить количество сотрудников, принятых на работу до 1992 года.

```
SELECT COUNT (*) AS N
     FROM EMPLOYEE
          WHERE HIRE_DATE < '01.01.1992';
```

Пример 12. Определить средний оклад сотрудников фирмы.

```
SELECT AVG (SALARY) AS AVG_SALARY
     FROM EMPLOYEE;
```

Задачи для самостоятельного выполнения

1. Выдать информацию обо всех отделениях, расположенных в `Monterey`.
2. Выдать информацию обо всех отделениях, входящих в отдел номер 110.
3. Получить список вакансий работы, предлагаемых в Японии.
4. Найти сотрудников, которые поступили на работу с 1992 по 1995 год.
5. Выдать список работ, для которых максимальная оплата ниже 150000.
6. Выдать список заказов, сидка на которые больше 20%.
7. Найти сотрудников, у которых оклад от 100000 до 150000, упорядочив его по дате поступления на работу.
8. Вывести список всех менеджеров среднего звена, если считать, что годовой доход менеджеров среднего звена лежит в пределах от 50000 до 80000.
9. Для заданного проекта вычислить его суммарный бюджет в 1995 году.
10. Вычислить суммарную стоимость заказов, сделанных в 1992 году.
11. Для отдела 123 вычислить сумму зарплат сотрудников этого отдела.
12. Выдать среднюю, минимальную и максимальную зарплату сотрудников указанного отдела (125).
13. Выдать список стран, в которых сотрудники указанной профессии (Eng) получают более 100000, упорядочив по алфавиту.

2.2 Запросы с группировкой данных

Предложение `GROUP BY` оператора `SELECT` позволяет объединить строки получающейся в результате запроса таблицы в группы и выдать для каждой

полученной группы обобщающую информацию. Группа формируется путем объединения всех строк, для которых столбец (или группа столбцов), указанный в предложении GROUP BY, имеет одинаковое значение. Столбец или группа столбцов, указанные в предложении GROUP BY, называются элементами группировки.

На запросы, в которых используется группировка, накладываются дополнительные ограничения:

- в качестве элементов группировки могут быть использованы только столбцы таблицы или выражения, вычисляемые на основе значений в столбцах таблицы;
- столбцами таблицы, формируемой оператором SELECT, могут быть или элементы группировки, или константы, или выражения с использованием агрегатных функций для столбцов, не являющихся элементами группировки.

Агрегатными функциями являются SUM(), MIN(), MAX() и AVG(). В качестве аргумента в этих функциях может быть использован столбец или выражение, содержащее столбец, не являющийся элементом группировки. Функции SUM() и AVG() применимы только к столбцам, содержащим числовые данные. Функции MIN() и MAX(), кроме числовых значений могут обрабатывать данные типа «дата–время». При вычислении значений агрегатных функций отсутствующие значения (null–значения) игнорируются.

Функция COUNT() в группирующих запросах также ведет себя как агрегатная функция. В случае использования COUNT(*) функция возвращает количество записей в группе. Если в качестве параметра функции COUNT используется имя столбца, не являющегося элементом группировки, функция вернет количество непустых значений в этом столбце для каждой из групп. Наконец, если совместно с указание имени столбца использовать DISTINCT, функция

COUNT() вернет количество различных значений в группе для указанного столбца.

При использовании группировки можно наложить дополнительное условие на результирующую таблицу, какие группы включать. Это условие задается предложением HAVING. Условие проверяется после того, как сформированы группы. Его рекомендуется применять, если условие отбора задается с помощью агрегатной функции.

Пример 13. Определить суммарный бюджет каждого проекта в 1994 году.

```
SELECT PROJ_ID, SUM (PROJECTED_BUDGET) AS TOTAL_BUDGET
     FROM PROJ_DEPT_BUDGET
     WHERE FISCAL_YEAR=1994
     GROUP BY PROJ_ID;
```

Пример 14. Определить средний бюджет каждого проекта в 1995 году, указав количество отделов, принимающих участие в проекте.

```
SELECT PROJ_ID,
     AVG (PROJECTED_BUDGET) AS TOTAL_BUDGET,
     COUNT (DEPT_NO) AS NUM_DEPARTMENTS
     FROM PROJ_DEPT_BUDGET
     WHERE FISCAL_YEAR=1995
     GROUP BY PROJ_ID;
```

Пример 15. Определить проекты, суммарный бюджет которых в 1994 году превысил 100000.

```
SELECT PROJ_ID, SUM (PROJECTED_BUDGET) AS TOTAL_BUDGET
     FROM PROJ_DEPT_BUDGET
     WHERE FISCAL_YEAR=1994
     GROUP BY PROJ_ID
   HAVING SUM (PROJECTED_BUDGET)>100000;
```

Пример 16. Для каждого кода работы определить в скольких странах предлагаются вакансии по этому виду работы.

```
SELECT JOB_CODE, COUNT (DISTINCT JOB_COUNTRY)
     FROM JOB
     GROUP BY JOB_CODE;
```

Задачи для самостоятельного выполнения

14. Для каждой страны определить количество сотрудников фирмы, работающих в этой стране.

15. Для каждой страны определить количество заказчиков, находящихся в этой стране.

16. Для каждого города определить, сколько отделений/филиалов фирмы расположено в этом городе.

17. Для каждого года определить, сколько сотрудников было принято в этом году на работу в фирму.

18. Для каждого года и каждой должности определить, сколько сотрудников было принято году на работу в фирму на эту должность и в этом.

19. Для каждого года и каждой страны определить, сколько сотрудников было принято в этом году и в этой стране на работу в фирму.

20. Для каждого года определить, сколько инженеров было принято на работу в этом году.

21. Для каждой страны определить, сколько администраторов работает в отделениях фирмы в этой стране.

22. Вывести суммарный бюджет всех проектов для каждого отдела за конкретный год.

23. Для каждого отдела вычислить суммарную зарплату сотрудников.

2.3 Получение данных из нескольких таблиц

На практике необходимо, чтобы запросы считывали данные сразу из нескольких таблиц.

SQL позволяет обратиться к нескольким таблицам в одном операторе SELECT, используя *соединения* (joins), *объединения* (unions) или *подзапросы* (subqueries), а также их любые комбинации.

2.3.1 Соединения

Соединение используется в операторе SELECT для получения ненормализованной таблицы, содержащей столбцы из нескольких исходных (соединяемых) таблиц, в которых хранятся логически связанные данные. Множество столбцов, выбранных из каждой таблицы, называется *потоком*.

Правило соединения строк из нескольких таблиц в одну строку задается условием соединения. Соединения бывают *внутренние* и *внешние*. *Внутреннее соединение* объединяет два потока таким образом, что несоответствующие условию соединения строки в любом из потоков отбрасываются. *Внешнее соединение* выбирает строки участвующих таблиц, даже если в некоторых случаях не найдено соответствие. При этом отсутствующие элементы данных в соединении определяются как NULL. В каждом внешнем соединении различают левую и правую таблицу. В зависимости от того строки какой из таблиц будут включены в результирующую таблицу, даже если для них не найдено соответствие, различают левое, правое и полное внешнее соединение.

В соединении могут участвовать не только разные таблицы. Возможно соединение таблицы самой с собой.

Полный синтаксис описания соединения:

```
FROM <таблица> [<алиас>]{[INNER]|{LEFT|RIGHT|FULL}[OUTER]} JOIN
    <таблица> [<алиас>]{[ON <условие соединения>[<соединение>]
```

Если вид соединения не указан, то соединение – внутреннее, для внешнего соединения достаточно указать является ли оно левым, правым или полным.

При использовании операции соединения может возникнуть коллизия имен столбцов (столбцы в разных таблицах имеют одинаковые имена). На уровне СУБД коллизия невозможна, т.к. полное имя столбца состоит из имени таблицы и имени столбца, разделенных точкой – это полный синтаксис именования столбца. Сокращенный синтаксис (указание только имени столбца) возможен, только если не возникает коллизия. Для упрощения написания полного имени столбца в операторе `SELECT` используется временное имя таблицы – *алиас*, он указывается непосредственно за именем таблицы в предложении `FROM` и может быть использован всюду в операторе `SELECT`.

Пример 17. Выдать список менеджеров отделов с указанием названия отдела.

```
SELECT FULL_NAME, DEPARTMENT
     FROM DEPARTMENT A JOIN
          EMPLOYEE B ON A.MNGR_NO =B.EMP_NO;
```

Так как в некоторых отделах нет менеджеров (см. пример 3), эти отделы не будут участвовать в соединении, они не попадут в результат запроса.

Пример 18. Выдать список ***всех*** отделов, указав полные имена управляющих (менеджеров).

```
SELECT DEPARTMENT, FULL_NAME
     FROM DEPARTMENT A LEFT JOIN
          EMPLOYEE B ON A.MNGR_NO =B.EMP_NO;
```

Или, поменяв местами таблицы в операции соединения

```
SELECT DEPARTMENT, FULL_NAME
     FROM EMPLOYEE A RIGHT JOIN
          DEPARTMENT B ON A.EMP_NO=B.MNGR_NO;
```

Поскольку таблица DEPARTMENT участвует во внешнем соединении, в результат запроса попадут строки обо всех отделах, но для тех из них, у которых нет менеджера, в столбце FULL_NAME будет указано значение null.

Пример 19. Выдать список сотрудников, указав названия проектов, в которых они участвуют.

```
SELECT FULL_NAME, PROJ_NAME
          FROM EMPLOYEE A JOIN EMPLOYEE_PROJECT B
                        ON A.EMP_NO=B.EMP_NO
                        JOIN PROJECT C ON B.PROJ_ID=C.PROJ_ID;
```

Пример 20. Для тех отделений, которые входят в состав более крупных, указать названия головного отдела.

```
SELECT A.DEPARTMENT, B.DEPARTMENT AS HEAD_DEPT
     FROM DEPARTMENT A
          JOIN DEPARTMENT B
               ON A.HEAD_DEPT=B.DEPT_NO;
```

Пример 21. Выдать список всех отделений, указав для тех, которые входят в состав более крупных, названия головного отдела.

```
SELECT A.DEPARTMENT, B.DEPARTMENT AS HEAD_DEPT
     FROM DEPARTMENT A
          LEFT JOIN DEPARTMENT B
               ON A.HEAD_DEPT=B.DEPT_NO;
```

Пример 22. Выдать список пар тех сотрудников из USA, которые заняты на одинаковой работе (совпадают job_code и job_grade).

```
SELECT A.FULL_NAME, B.FULL_NAME, A.JOB_CODE, A.JOB_GRADE
          FROM EMPLOYEE A JOIN EMPLOYEE B ON
          (A.JOB_CODE=B.JOB_CODE) AND (A.JOB_GRADE=B.JOB_GRADE)
          WHERE (A.JOB_COUNTRY='USA') AND (B.JOB_COUNTRY='USA')
          AND (A.EMP_NO<B.EMP_NO);
```

Задачи для самостоятельного выполнения

24. Выдать список сотрудников, оклад которых ниже 50000, указав наименование их работы.

25. Для каждого проекта выдать зарплату руководителя этого проекта

26. Вывести список отделов, зарплата начальников которых выше 80000.

27. Выдать список начальников отделов с указанием их номера телефона и оклада.

28. Для каждого проекта указать страну, в которой находится руководитель проекта.

29. Выдать историю изменения оплаты начальника конкретного отдела.

30. Для каждого заказа выдать стоимость заказа в валюте страны, где расположен заказчик.

31. Для каждого заказа указать страну, в которой находится сотрудник, оформлявший договор-заказ.

32. Найти всех инженеров, работающих над указанным проектом

33. Выдать список сотрудников, работающих над данным проектом, с указанием их оклада, отсортировав список по убыванию оклада.

34. Выдать список сотрудников, клиенты которых (заказчики) живут в Америке, указав для них полный адрес.

35. Найти число сотрудников, работающих на должности, заданной по названию.

36. Вывести список отделов, зарплата начальников которых выше 80000.

37. Выдать список сотрудников, оклад которых ниже 60000, указав наименование их работы, наименование отдела и наименование проекта.

38. Выдать список клиентов, оплативших и получивших заказ в 1992 году.

39. Выдать список сотрудников, принятых на работу с 1993 года, с указанием окладов и отделов, в которых они работают, отсортировать список по убыванию оклада.

40. Найти сотрудников, получающих оплату в фунтах (`Pound`).

41. Выдать список заказчиков из USA и общую сумму заказов.

42. Выдать список сотрудников, оформлявших заказы с клиентом ‘`3D-Pad Corp.`’ с указанием их оклада, отсортировать список по убыванию оклада.

43. Выдать список руководителей проектов, указав сумму, на которую им повысили зарплату, а если не повысили, то оставить поле пустым.

44. Найти все проекты каждого отдела.

45. Вывести средний процент повышения оклада работников данного проекта.

46. Выдать список сотрудников, чья зарплата была повышена на 7-9%, с указанием полного имени, должности, новой и старой зарплаты.

47. Найти отделы, в которых у сотрудников было повышение оклада больше чем на 10000.

48. Для начальников отделов выдать диапазон оплаты для выполняемой ими работы.

49. Вывести дату последнего повышения оклада сотрудникам данного проекта.

2.3.2 Объединения

Оператор `UNION` используется для объединения результатов двух или более операторов `SELECT`, результаты которых должны быть *совместимы для объединения*. Для каждого оператора `SELECT`, передающего выходной поток оператору `UNION`, спецификация должна содержать список одинаковых столбцов (количество, порядок и типы соответствующих столбцов должны совпадать).

Если в процессе создания объединенного набора были сформированы дублирующие строки, то в результирующий набор попадет только один экземпляр. Для включения дубликатов нужно указать UNION ALL.

Пример 23. Выдать полный список стран, вовлеченных в деятельность фирмы.

```
SELECT DISTINCT JOB_COUNTRY
     FROM JOB
UNION
SELECT DISTINCT COUNTRY
     FROM CUSTOMER;
```

Пример 24. Выдать список всех менеджеров отделов и всех руководителей проектов.

```
SELECT FULL_NAME,'MANAGER' AS TITLE
     FROM EMPLOYEE E JOIN DEPARTMENT D
          ON E.EMP_NO=D.MNGR_NO
UNION
SELECT FULL_NAME,'TEAM_LEADER'
     FROM EMPLOYEE F JOIN PROJECT P
          ON F.EMP_NO=P.TEAM_LEADER
```

Задачи для самостоятельного выполнения

50. Выдать список всех менеджеров отделов и всех руководителей проектов из USA, указав их оклады.

51. Выдать список всех филиалов фирмы из USA и всех заказчиков из USA.

52. Для каждой страны указать, сколько в ней имеется филиалов фирмы и сколько заказчиков расположены в этой стране.

2.3.3 Подзапросы

Подзапрос – это оператор SELECT, включенный в спецификацию другого оператора SQL.

В Firebird версии 1.5 и выше выражение подзапроса может быть использовано тремя способами:

– в операторе INSERT для получения данных, вставляемых из одной таблицы в другую;

– для формирования выходного столбца основного оператора SELECT;

– для получения значений или условий, используемых в предикатах поиска предложения WHERE операторов SELECT, UPDATE и DELETE, а также в предложении HAVING для группирующего запроса.

Начиная с версии Firebird 2.0, появилась возможность использовать подзапрос в предложении FROM. В этом случае подзапрос выступает в роли «производной таблицы» (*derived table*). Синтаксис такого подзапроса несколько отличается от обычного синтаксиса, т.к. производной таблице можно дать имя, имена можно дать также столбцам производной таблицы:

```
(select-запрос)
[[AS] алиас производной таблицы]
[(<псевдонимы полей производной таблицы>)]
```

Если подзапрос ссылается на столбцы в таблицах включающего его запроса, то он называется *коррелированным*.

Следует отметить, что в большинстве случаев подзапрос является альтернативной формой решения, которое может быть получено с помощью соединения.

Пример 25. Выдать список сотрудников с указанием оклада и валюты, в которой они получают оплату.

```
SELECT FULL_NAME, SALARY,
     (SELECT CURRENCY FROM COUNTRY
          WHERE COUNTRY.COUNTRY=EMPLOYEE.JOB_COUNTRY) AS CURRENCY
     FROM EMPLOYEE;
```

Используемый подзапрос является коррелированным. Очевидно, что вместо подзапроса можно было использовать соединение.

Пример 26. Выдать список сотрудников, работающих в США с указанием их оклада. Привести для справки максимальную оплату для сотрудников в США.

```
SELECT e.FULL_NAME, e.SALARY,
     (SELECT MAX (m.SALARY)
          FROM EMPLOYEE m
          WHERE m.JOB_COUNTRY='USA') AS MAX_SALARY_USA
     FROM EMPLOYEE e
     WHERE e.JOB_COUNTRY='USA'
```

В данном случае подзапрос является некоррелированным.

Пример 27. Для каждого проекта указать, какие отделы принимали участие в его выполнении в 1995 году. Для проектов, которые не выполнялись в 1995 году вместо названия отдела указать `null`.

```
SELECT  P.PROJ_NAME, T.DEPT_NAME
FROM PROJECT P LEFT JOIN
     (SELECT PD.PROJ_ID,D.DEPARTMENT
          FROM PROJ_DEPT_BUDGET PD JOIN DEPARTMENT D
                    ON (PD.DEPT_NO=D.DEPT_NO)
          WHERE PD.FISCAL_YEAR=1995)
     AS T ( PROJ_ID, DEPT_NAME)
     ON(P.PROJ_ID=T.PROJ_ID)
```

Пример 28. Получить список сотрудников, имеющих оклад выше среднего по фирме.

```
SELECT FULL_NAME
       FROM EMPLOYEE a
       WHERE a.SALARY >(SELECT AVG(b.SALARY)
                    FROM EMPLOYEE b);
```

Пример 29. Выдать названия отделов, бюджет которых, совпадает с бюджетом отдела номер 130.

```
SELECT a.DEPARTMENT
     FROM DEPARTMENT a
     WHERE a.BUDGET = (SELECT b.BUDGET
                              FROM DEPARTMENT b
                              WHERE b.DEPT_NO=130)
          AND NOT a.DEPT_NO=130;
```

Пример 30. Определить проекты, для которых максимальный бюджет, выделявшийся для одного отдела в 1994 году, был выше среднего бюджета по всем отделам и проектам в этом году.

```
SELECT PROJ_ID, MAX(PROJECTED_BUDGET) AS TOTAL_BUDGET
         FROM PROJ_DEPT_BUDGET
         WHERE FISCAL_YEAR=1994
         GROUP BY PROJ_ID
     HAVING MAX(PROJECTED_BUDGET)>
          (SELECT AVG(a.PROJECTED_BUDGET)
               FROM PROJ_DEPT_BUDGET a
               WHERE a.FISCAL_YEAR=1994);
```

Задачи для самостоятельного выполнения

53. . Выдать список сотрудников, работающих в Канаде и Англии с указанием их оклада. Привести для справки максимальную и минимальную оплату для сотрудников в этих странах.

54. Для каждого проекта указать, какие отделы принимали участие в его выполнении в 1994 году и руководителей этих отделов. Для проектов, которые не выполнялись в 1994 году вместо названия отдела указать `null`.

55. Определить проекты, для которых максимальный бюджет, выделявшийся для одного отдела в 1994 году, был ниже среднего бюджета по всем отделам и проектам в этом году.

56. Выдать названия отделов, бюджет которых, выше бюджета отдела номер 130.

3 ВЫРАЖЕНИЯ И ПРЕДИКАТЫ

Выражения могут быть использованы, если информация, помещаемая в таблицы базы данных или выбираемая из базы данных, должна быть подвергнута некоторой обработке.

Операндами в выражениях могут быть имена столбцов, константы, внутренние контекстные переменные и литералы даты, а также другие выражения.

В качестве операций могут использоваться арифметические операции, логические операции, операции сравнения, операция конкатенации, предикаты существования, предикаты отсутствия значения, вызовы функций.

Порядок вычисления выражения определяется приоритетами операций и может быть изменен с помощью круглых скобок.

При построении выражений следует учитывать, что результат вычисления может зависеть от отсутствия значений операндов (`null`- значений). Большинство СУБД, в случае отсутствия значения одного из операндов, полагает результат равным `null`.

Предикаты – это логические выражения. Предикаты позволяют записывать условия, поверка истинности которых влияет на выполнение операторов SQL. Предикат может быть истинным, ложным и неопределенным (в случае, когда хотя бы один из операндов предиката имеет значение `null`). В SQL лож-

ный и неопределенный результат объединяются и трактуются как ложь. Простейшие предикаты строятся с помощью операций сравнения (=, <, >, < >, >=, <=) и логических операций NOT, AND, OR.

Ниже будут рассмотрены только те элементы выражений, допустимые в реализации SQL СУБД Firebird, которые имеют особенности использования по сравнению с традиционными языками программирования.

3.1 Оператор конкатенации

Оператор конкатенации (||) создает новую строку путем соединения двух строк. Операция конкатенации проверяет результирующую строку на допустимую длину. Если длина превышена, никакого усечения строки не происходит и выдается сообщение об ошибке.

Соединяемые стоки могут быть константами или значениями, полученными из таблицы, или значениями, возвращаемыми строковыми функциями.

Если хотя бы один из операндов не содержит значения (null), то и результирующая строка будет null.

3.2 Операторы сравнения и предикаты

При использовании операций сравнения на равенство и неравенство следует учитывать, что если хотя бы один из операндов, участвующих в сравнении имеет значение null, то операция сравнения вернет значение «не определено». В логических выражениях оператора SQL это значение будет трактоваться как «ложь».

Наряду с традиционными операциями сравнения, в SQL используются дополнительные операции, которые упрощают форму записи сложных условий.

Предикаты IS NULL и IS NOT NULL проверяют, что объект в левой части операции сравнения не имеет (или имеет) значение. Если значение имеется,

то предикат IS NULL вернет «ложь», а IS NOT NULL вернет «истину». И наоборот.

Предикат BETWEEN…AND… проверяет, что значение попадает в диапазон значений.

Предикат CONTAINING проверяет, содержится ли указанная последовательность символов в строковом представлении операнда (без учета регистра).

Предикат IN (…) проверяет, присутствует ли значение в указанном списке. Список может быть составлен явно из константных значений или быть результатом выполнения подзапроса.

Предикат LIKE позволяет создавать чувствительный к регистру шаблон поиска в символьной строке. Он позволяет использовать два шаблонных символа: % - для представления любого количества любых символов; символ подчеркивания _ для представления одного неопределенного символа.

```
LIKE '%mit%'
LIKE 'Sm_th'
```

Предикат STARTING WITH проверяет, что строка символов начинается с указанной последовательности символов с учетом регистра.

```
STARTING WITH 'Jo'
```

Пример 31. Выдать информацию об отделах, не имеющих управляющего.

```
SELECT * FROM DEPARTMENT WHERE MNGR_NO IS NULL;
```

Пример 32. Выдать сведения о сотрудниках, принятых на работу в 1992 году.

```
SELECT * FROM EMPLOYEE
     WHERE HIRE_DATE BETWEEN '01.01.1992' AND '31.12.1992';
```

Пример 33. Выдать сведения обо всех изменениях оплаты за 93 год.

```
SELECT * FROM SALARY_HISTORY
     WHERE CHANGE_DATE CONTAINING 93;
```

Пример 34. Выбрать сведения о сотрудниках с указанными именами.

```
SELECT * FROM EMPLOYEE
     WHERE FIRST_NAME IN ('Pete','Ann','Roger');
```

Пример 35. Получить список заказчиков, с которыми был оформлен договор 12 декабря 1993 года.

```
SELECT CUSTOMER
     FROM CUSTOMER
     WHERE CUST_NO IN (SELECT CUST_NO
                              FROM SALES
                              WHERE ORDER_DATE='12.12.1993');
```

Пример **36.** Выдать список служащих тех отделов, у которых нет менеджера.

```
SELECT FULL_NAME FROM EMPLOYEE
     WHERE DEPT_NO IN (
          SELECT DEPT_NO
          FROM DEPARTMENT WHERE MNGR_NO IS NULL);
```

Пример 37. Выдать список заказчиков – корпораций.

```
SELECT C.CUSTOMER
     FROM CUSTOMER C
     WHERE C.CUSTOMER LIKE '%Corp%'
```

Пример 38. Выдать список сотрудников, у которых имя начинается с буквы R.

```
SELECT FULL_NAME
     FROM EMPLOYEE
     WHERE FIRST_NAME STARTING WITH 'R'
```

3.3 Предикаты существования

Предикаты существования используют подзапросы для передачи значений для различного виды утверждений в условиях поиска. Все они проверяют существование значения в левой части предиката в выходных результатах подзапроса.

Предикат ALL проверяет, является ли сравнение (обычно в виде неравенства) истинным для всех значений, возвращаемых подзапросом.

Предикат EXISTS (NOT EXISTS) определяет, существует ли (или нет), по крайней мере, одно значение в выходном результате подзапроса. В большинстве случаев подзапросы в предикате EXISTS являются *коррелированными,* то есть условия поиска подзапроса связаны с одним или более столбцами основного запроса. Так как смысл этого предиката – проверить существование кортежей в подзапросе, сам подзапрос может иметь любой список возвращаемых столбцов, в частности это может быть один столбец, содержащий любую константу.

Предикат SINGULAR (NOT SINGULAR) проверяет, возвращает ли подзапрос в точности одно значение. Если возвращается NULL или более одного значения, то SINGULAR возвращает ложь.

Предикаты SOME и ANY проверяют, является ли сравнение истинным, по крайней мере, для одного значения, возвращаемого подзапросом. Применяются для сравнения на неравенство или проверке условий с помощью LIKE, CONTAINING и т.д. При сравнении на равенство они эквивалентны предикату EXISTS.

Пример 39. Найти проекты, годовой бюджет которых превышал бюджет любого отдела.

```
SELECT PROJ_ID, DEPT_NO, FISCAL_YEAR
     FROM PROJ_DEPT_BUDGET
     WHERE PROJECTED_BUDGET> ALL(SELECT BUDGET FROM DEPARTMENT);
```

Пример 40. Найти сотрудников отдела 130, которые оформляли заказы.

```
SELECT FULL_NAME
     FROM EMPLOYEE e
     WHERE DEPT_NO =130
          AND EXISTS (SELECT * FROM SALES s
                         WHERE s.SALES_REP=e.EMP_NO);
```

Пример 41. Найти сотрудников отдела 130, которые не никогда оформляли заказы.

```
SELECT FULL_NAME FROM EMPLOYEE e
     WHERE DEPT_NO =130
          AND NOT EXISTS (SELECT * FROM SALES s
                         WHERE s.SALES_REP=e.EMP_NO);
```

Пример 42. Выдать список сотрудников, которые являются руководителями проектов.

```
SELECT a.FULL_NAME
     FROM EMPLOYEE a
     WHERE EXISTS(SELECT 1
                         FROM PROJECT p
                         WHERE p.TEAM_LEADER = a.EMP_NO);
```

Пример 43. Выдать список всех сотрудников, указав, кто из них является, а кто – нет, руководителем проекта.

```
SELECT a.FULL_NAME, 'YES' as "TEAM LEADER?"
     FROM EMPLOYEE a
     WHERE EXISTS (SELECT 1 FROM PROJECT p
                         WHERE p.TEAM_LEADER = a.EMP_NO)
UNION
SELECT b.FULL_NAME, 'NO ' as "TEAM LEADER?"
     FROM EMPLOYEE b
     WHERE NOT EXISTS(SELECT 1 FROM PROJECT t
                         WHERE t.TEAM_LEADER = b.EMP_NO);
```

Пример 44. Найти сотрудников, которые оформили только один заказ.

```
SELECT FULL_NAME
     FROM EMPLOYEE e
     WHERE SINGULAR (SELECT * FROM SALES s
               WHERE s.SALES_REP=e.EMP_NO);
```

Пример 45. Выдать список всех служащих, у которых было, по крайней мере, одно изменение оклада в течение года после приема на работу.

```
SELECT e.EMP_NO, e.FULL_NAME, e.HIRE_DATE
     FROM EMPLOYEE e
     WHERE e.HIRE_DATE+365>
          SOME(SELECT sh.CHANGE_DATE
                    FROM SALARY_HISTORY sh
                    WHERE sh.EMP_NO = e.EMP_NO);
```

3.4 Выражение CASE

Выражение CASE позволяет определить значение результирующего столбца зависимым от результата вычисления группы взаимоисключающих условий.

```
CASE {<выражение 1>|<пустое предложение>}
     WHEN {{NULL|<значение 1>}|<предикат поиска>}
          THEN {<результат 1>|NULL}
     WHEN . . .THEN {<результат 2>|NULL}
     [WHEN . . .THEN {<результат n>|NULL}]
     [ELSE {<результат n+1>|NULL}]
END
```

Фраза WHEN . . .THEN является ключевыми словами в каждом предложении, связывающем условие и результат. Требуется, по крайней мере, одно предложение условие/результат. Фраза ELSE предшествует необязательному последнему результату, который возвращается, если не выполняется ни одно предыдущее предложение.

Аргумент ***выражение 1*** является выражением, значение которого определяет результат предложения CASE. Если оно указано, то результат выражения будет последовательно сравниваться со значениями указанными в предложениях WHEN. Если ***выражение 1*** опущено, то в предложении WHEN должны указываться предикаты.

Предложение CASE возвращает единственное значение. Если не выполняется ни одно из условий и не указано предложение ELSE, то возвращаемый результат будет NULL.

Пример 46. В таблице «Заказы» (ORDERS) указывается состояние, в котором находится заказ (status): 1 – подтвержден, 2 – выполняется, 3 – готов, 4 – исполнен. Пока заказ не находится ни в одном из состояний, в поле status находится значение null. Выдать список заказов, указав для каждого, в каком состоянии он находится.

```
SELECT o.ID,
 CASE o.STATUS
     WHEN 1 THEN 'confirmed'
     WHEN 2 THEN 'in production'
     WHEN 3 THEN 'ready'
     WHEN 4 THEN 'shipped'
     ELSE 'unknown status'||o.STATUS
END
FROM ORDERS o;
```

Вариант синтаксиса, использующего предикаты

```
SELECT o.ID,
 CASE
     WHEN (o.STATUS IS NULL) THEN 'new'
     WHEN (o.STATUS=1) THEN 'confirmed'
     WHEN (o.STATUS=2) THEN 'in production'
     WHEN (o.STATUS=3) THEN 'ready'
     WHEN (o.STATUS=4) THEN 'shipped'
     ELSE 'unknown status'||o.STATUS
END
FROM ORDERS o;
```

3.5 Функции

Функции могут использоваться при построении выражений.

Функция CAST() позволяет преобразовывать элемент данных одного типа данных в другой тип.

```
CAST(<значение> AS <тип данных>)
```

Функция EXTRACT() позволяет выделять часть информации из полей типа DATE, TIME и TIMESTAMP. Результат возвращается в виде числа.

```
EXTRACT (<часть> FROM <поле>)
```

Выделяемая часть определяется следующими ключевыми словами:

```
YEAR | MONTH | DAY | HOUR | MINUTE | SECOND | WEEKDAY| YEARDAY
```

Функция SUBSTRING() возвращает подстроку, начиная с указанной позиции.

```
SUBSTRING (<строка> FROM <начальная позиция> [FOR <длина>])
```

Функция UPPER() преобразует все символы строки в верхний регистр. Противоположная ей функция LOWER() появилась в версии Firebird 2.0.

```
UPPER(<строка>)
LOWER(<строка>)
```

Функции CHAR_LENGTH() и BIT_LENGTH() возвращают длину строки в символах/битах соответственно (появились в версии Firebird 2.0).

```
CHAR_LENGTH(<строка>)
BIT_LENGTH(<строка>)
```

Функция TRIM() удаляет начальные и концевые пробелы в строке (появилась в версии Firebird 2.0).

```
TRIM(<строка>)
```

Функция GEN_ID() вычисляет и возвращает очередное значение указанного генератора. Использование генераторов больше связано с программированием процедур и триггеров.

```
GEN_ID(<генератор>,<шаг>)
```

Агрегатные функции чаще всего используются в комбинации группировкой для вычисления итогов для групп. Агрегатными функциями являются SUM(), MAX(), MIN(), AVG(), COUNT(). При вычислении агрегатных функций значения NULL не учитываются.

Функция COALESCE() позволяет вычислить значение результата с использованием серии выражений. Результат будет равен значению первого выражения в списке, которое окажется непустым.

```
COALESCE (<выражение 1> {,<выражение 2>[,…<выражение n>]})
```

Функция NULLIF() сравнивает значения двух аргументов и возвращает NULL, если они равны, или первый аргумент, если они не равны.

```
NULLIF(<выражение 1>,<выражение 2>)
```

Функция IIF() принимает на вход три параметра. Если первый параметр равен TRUE, функция возвращает значение второго параметра, иначе возвращает значение третьего параметра (появилась в версии Firebird 2.0)..

```
IIF (<условие>, результат1, результат2)
```

Пример 47. Найти заказчиков, в названии которых есть аббревиатура ‘Ltd’. Анализ таблицы CUSTOMER показывает, что при написании заглавные и малые буквы используются по-разному.

```
SELECT c.CUSTOMER
    FROM CUSTOMER c
    WHERE UPPER(c.CUSTOMER) CONTAINING 'LTD';
```

Пример 48. Выдать список сотрудников, принятых на работу в 1992 году.

```
SELECT *
    FROM EMPLOYEE s
    WHERE EXTRACT(YEAR FROM s.HIRE_DATE)=1992;
```

Пример 49. Выдать справочник телефонов сотрудников. Телефон сотрудника состоит из телефона отдела, в котором работает сотрудник и необязательного добавочного кода.

```
SELECT a.FULL_NAME, d.PHONE_NO || COALESCE('+'||a.PHONE_EXT,' ')
     FROM EMPLOYEE a JOIN DEPARTMENT d ON (a.DEPT_NO=d.DEPT_NO);
```

Пример 50. Для каждого сотрудника указать, работает ли он более 20 полных лет или нет.

```
SELECT m.FULL_NAME, m.HIRE_DATE ,
     IIF(EXTRACT(YEAR FROM CURRENT_DATE)-
          EXTRACT(YEAR FROM m.HIRE_DATE)>20,'Более 20','Менее 20')
     FROM EMPLOYEE m;
```

Пример 51. Для каждого сотрудника указать, кто из них работает более 20 лет, более 15 лет и остальные

```
SELECT m.FULL_NAME, m.HIRE_DATE ,
   CASE
     WHEN
          EXTRACT(YEAR FROM CURRENT_DATE)-
          EXTRACT(YEAR FROM m.HIRE_DATE)>20
     THEN 'Более 20'
     WHEN
          EXTRACT(YEAR FROM CURRENT_DATE)-
          EXTRACT(YEAR FROM m.HIRE_DATE)>15
     THEN 'Более 15'
     ELSE 'Менее 15'
   END
     FROM EMPLOYEE m;
```

Задачи для самостоятельного выполнения

57. Найти проекты, в названии которых встречается строка ‘Map’.
58. Найти сотрудников, у которых фамилия начинается на ‘Jo’.
59. Вывести список сотрудников, фамилии которых оканчиваются на ‘son’.
60. Найти отделы, в названии которых встречается строка ‘Office’.
61. Найти клиентов (заказчиков), название которых начинается с ‘Dyn’.

62. Выдать отделы, международный код телефона которых начинается на 8.

63. Найти отделы, в названии которых встречается строка ‘Software’

64. Найти названия проектов, начинающихся с ‘Ma’.

65. Для каждой работы указать в одной колонке диапазон оплаты с добавлением наименования валюты страны, где предоставляется работа.

66. Для данного отдела найти все проекты, в которых сотрудники отдела являются руководителями, указав их полное имя и наименование работы.

67. Выдать список проектов, в которых участвуют начальники отделов.

68. Найти отделы, руководители которых являются и руководителями проектов, указав название проекта.

69. Выдать список сотрудников, работающих над данным проектом, с указанием их оклада, наименования должности и добавив столбец, в котором для руководителя проекта будет указано – ‘header’, а для остальных сотрудников – пусто. Список отсортировать по убыванию оклада.

70. Вывести список сотрудников из USA, оплата у которых выше средней оплаты по той работе, которую они выполняют.

71. Вывести список сотрудников и изменение их зарплаты, начиная с 1993 года. Добавить столбец, в котором указать ‘Hight’ если зарплата повысилась, ‘Low’ – если понизилась.

72. Найти руководителей проектов, которым снизили оплату с указанием названия проекта.

4 ВСТАВКА СТРОК В ТАБЛИЦУ

Оператор INSERT используется для добавления строк в одну таблицу. Возможна вставка одной строки, для которой список добавляемых значений столбцов указывается явно в виде списка констант.

```
INSERT INTO <таблица> [(<список столбцов>)]
     VALUES (<список значений>)
```

Другая форма оператора INSERT использует в качестве вставляемых строк результат подзапроса. В этом случае одним оператором INSERT можно добавить более одной строки.

```
INSERT INTO <таблица> [(<список столбцов>)]
     (SELECT ...)
```

Пример 52. Вставить в таблицу COUNTRY новую строку.

```
INSERT INTO COUNTRY(COUNTRY,CURRENCY)
               ('RUSSIA', 'RUB');
```

Пример 53. Таблица SALES_ARH имеет структуру, полностью совпадающую со структурой таблицы SALES. Добавить в нее строки, содержащие информацию обо всех полностью закрытых заказах (поставка и оплата произведены).

```
INSERT INTO SALES_ARH
     SELECT *
          FROM SALES
          WHERE SHIP_DATE IS NOT NULL AND PAID='y';
```

5 ОБНОВЛЕНИЕ СТРОК В ТАБЛИЦЕ

Оператор UPDATE используется для изменения значений столбцов в существующих строках таблицы.

```
UPDATE <таблица>
     SET <столбец> = <значение> [,<столбец> = <значение> .. ]
     [WHERE <условие поиска> ]
```

Если не задано предложение WHERE, то изменения будут выполнены для каждой строки таблицы.

Пример 54. Установить сотруднику с номером 65 оплату в размере 40000.

```
UPDATE EMPLOYEE
     SET SALARY = 40000
     WHERE EMP_NO = 65;
```

Пример 55. Увеличить всем сотрудникам оплату на 1000.

```
UPDATE EMPLOYEE
          SET SALARY =SALARY+1000;
```

6 УДАЛЕНИЕ СТРОК ИЗ ТАБЛИЦЫ

Оператор DELETE используется для удаления строк таблицы

```
DELETE FROM <таблица>
     [WHERE <условие поиска>]
```

Если не указано предложение WHERE, то будут удалены все строки таблицы.

Пример 56. Удалить из списка стран строку, относящуюся к России.

```
DELETE FROM COUNTRY
     WHERE COUNTRY = 'RUSSIA';
```

Пример 57. Удалить из таблицы SALARY_HISTORY все сведения об изменении оплаты сотруднику с номером 65.

```
DELETE FROM SALARY_HISTORY
     WHERE EMP_NO = 65;
```

Пример 58. Удалить из таблицы SALES строки, содержащие информацию обо всех полностью закрытых заказах (поставка и оплата произведены). Данные из этих строк перемещались в таблицу SALES_ARH (см. пример 53).

```
DELETE FROM SALES
     WHERE SHIP_DATE IS NOT NULL AND PAID='y';
```

Пример 59. Очистить таблицу SALES_ARH.

```
DELETE FROM SALES_ARH;
```

Пример 60. Данный пример не связан с демонстрационной базой данных. В нем показано решение довольно часто встречающейся задачи – удалить из таблицы дубликаты строк. Пусть имеется таблица TABL, содержащая два столбца ID и TEXT. В столбце TEXT могут встречаться повторяющиеся значе-

ния. В столбце ID все значения различные – этого можно добиться, вставив в таблицу дополнительный столбец. В некоторых СУБД в качестве такого столбца можно использовать системное значение номера строки.

ID	TEXT
1	abc
2	abc
3	3
4	abc
5	3
6	dsds
7	trtt

Необходимо удалить все строки, в которых содержатся повторяющиеся значения поля TEXT, оставив только одну – любую. Можно, например, оставить ту строку, в которой номер ID меньше.

```
DELETE FROM T1 A
  WHERE EXISTS (SELECT 1 FROM T1 B
                  WHERE A.TEXT=B.TEXT AND A.ID>B.ID);
```

Задачи для самостоятельного выполнения

73. Добавить в таблицу EMPLOYEE сведения о новом сотруднике. При заполнении данных о работе сотрудника и его окладе использовать сведения из таблицы JOB. При указании номера отдела использовать таблицу DEPARTMENT.

74. Включить сотрудника, добавленного в предыдущей задаче, в работу над одним из проектов. Для этого добавить запись в таблицу EMPLOYEE_PROJECT.

75. Изменить сотруднику, добавленному в задаче 73 оплату. При изменении учитывать ограничения по диапазону оплаты из таблицы JOB.

76. Удалить записи из таблиц EMPLOYEE_PROJECT и EMPLOYEE, добавленные при выполнении предыдущих задач.

ЛИТЕРАТУРА

1 Борри Х. Firebird: руководство разработчика баз данных.–СПб.:БХВ–Петербург, 2006.–1104с.

2 Гарсия–Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс.–М.:Издательский дом «Вильямс», 2004.-1088с.

3 Джеймс Р. Грофф, Пол Н. Вайнберг. SQL: полное руководство. – К.: Издательская группа BHV, 1998.

4 К.Дейт. Введение в системы баз данных. 7-е издание. : Пер. с англ. — М. : Издательский дом «Вильямс», 2000.–848с.

5 К.Дейт. Руководство по реляционной СУБД DB2. М.: ФиС, 1988.

6 Ковязин А., Востриков С. Мир InterBase. Архитектура, администрирование и разработка приложений баз данных в InterBase/Firebird/Yaffi.–М.:КУДИЦ-ОБРАЗ; СПб.:Питер,2005.–496с.

7 Моисеенко С.И. SQL. Задачи и решения.–СПб.:Питер, 2006. –255с.

8 Петров В.Н. Информационные системы: Учебник для вузов.- Изд-во «Питер», 2002.

9 Хансен Г., Хансен Дж.. Базы данных: разработка и управление.–М.:БИНОМ, 1999.

ПРИЛОЖЕНИЕ

ОПИСАНИЕ ДЕМОНСТРАЦИОННОЙ БАЗЫ ДАННЫХ

Примеры, приводимые в методических указаниях и задачи для самостоятельной работы, основаны на демонстрационной базе данных СОТРУДНИКИ (EMPLOYEE), устанавливаемой автоматически при инсталляции сервера СУБД Firebird. Аналогичные демонстрационные базы устанавливаются практически всеми реляционными СУБД, однако в каждом конкретном случае структура таблиц и хранящаяся в них информация могут быть различными.

На рисунке 1 приведена схема базы данных с указанием первичных и внешних ключей.

База данных employee.fdb содержит следующую информацию (в скобках указаны имена таблиц, в которых находится соответствующая информация и названия столбцов, содержащих соответствующую информацию):

- информация о сотрудниках(EMPLOYEE) некоторой фирмы, занимающейся разработкой проектов в сфере компьютерных технологий и электроники, а так же продажей своей продукции. Каждый сотрудник работает в каком-то отделе/филиале (dept_no). Каждый сотрудник имеет конкретную должность (job_code) и квалификацию (job_grade), работает в конкретной стране (job_country);
- фирма состоит из отделов (DEPARTMENT): финансового, маркетинга и других. В число отделов входят и филиалы по всему миру. Каждый филиал (отдел) подчинен вышестоящему отделу (head_dept). Так, например, филиал в Италии непосредственно подчинен Европейскому управлению фирмы, которое в свою очередь подчинено главному корпоративному управлению. Отделами руководят менеджеры из числа сотрудников (mng_no).;
- сотрудники приняты для выполнения определенной работы (JOB), вакансия по конкретной должности (job_code) предлагается в какой-то из

стран (job_country) и требует определенной квалификации (job_grade). По каждой вакансии могут быть установлены дополнительные требования к кандидату и зафиксирован диапазон оплаты. Диапазон оплаты указывается в валюте той страны, в которой предоставляется вакансия;

- каждый сотрудник может участвовать в выполнении одного или нескольких проектов (EMPLOYEE_PROJECT);
- каждым проектом, выполняемым фирмой (PROJECT), руководит один из сотрудников (team_leader);
- для каждого отдела, работающего над конкретным проектом, определяется годовой бюджет выполнения этого проекта (PROJ_DEPT_BUDGET);
- информация об изменении оплаты сотрудникам (SALARY_HISTORY) сохраняется за все время их работы;
- сотрудники могут оформлять заказы на поставку технического и программного обеспечения (SALES) фирмам–клиентам (CUSTOMER). Конкретным заказом занимается определенный торговый представитель (sales_rep) из числа сотрудников. Каждый заказ проходит стадии: оформление, исполнение, передача клиенту. Если заказ не оплачивается в течение двух месяцев, после того как он переходит в разряд поставленных, то сотрудничество с фирмой–клиентом замораживается;
- таблица COUNTRY является справочником стран, в которых расположены филиалы фирмы, сотрудники и клиенты, в ней содержится наименование основной валюты страны.

В приведенных выше описаниях имена таблиц указаны заглавными буквами, а имена полей – строчными. Следует отметить, что и стандарт SQL и большинство реализаций языка SQL безразличны к регистру символов при написании операторов SQL, указании имен таблиц и полей.

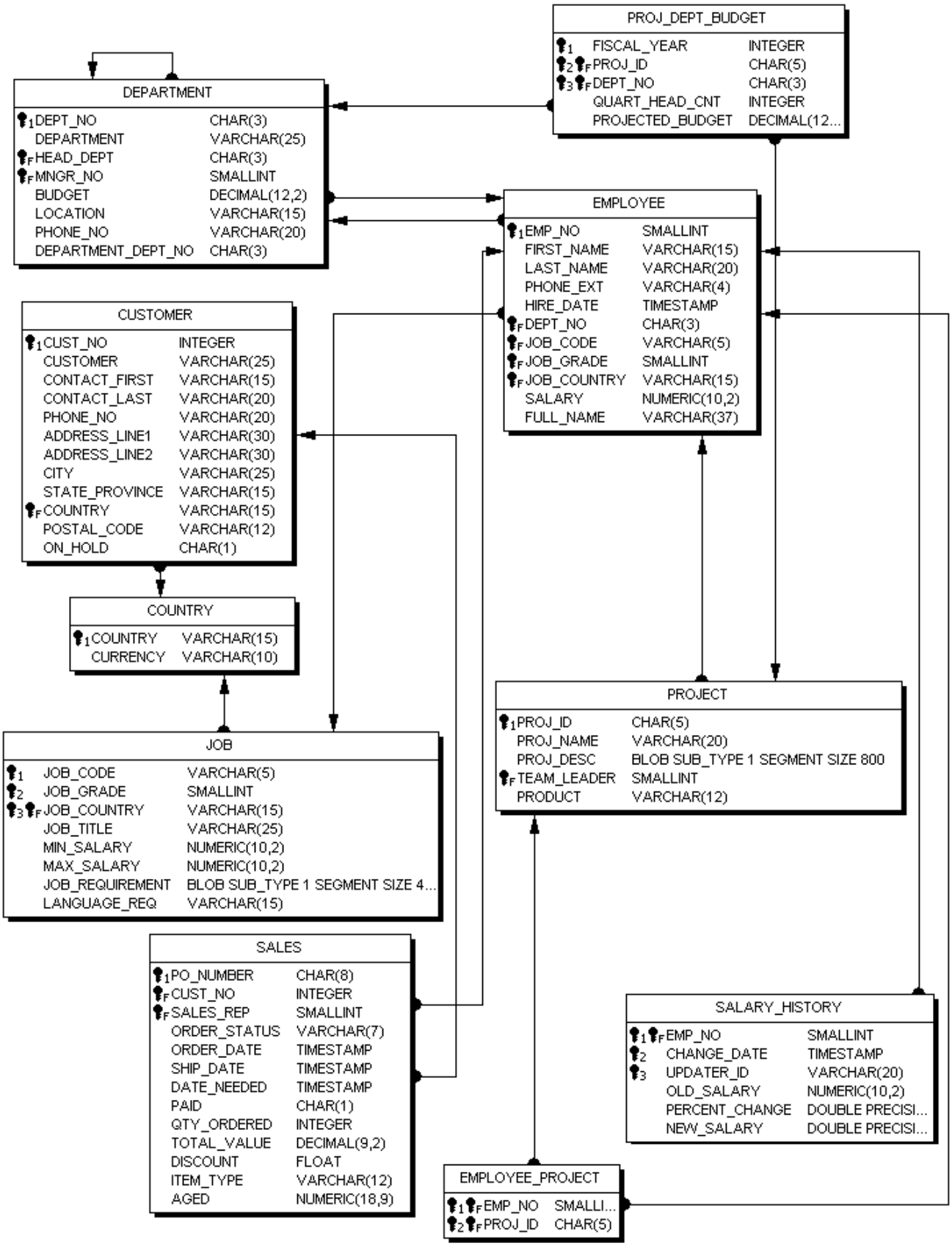

Рисунок 1. Схема базы данных employee.fdb