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.

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.

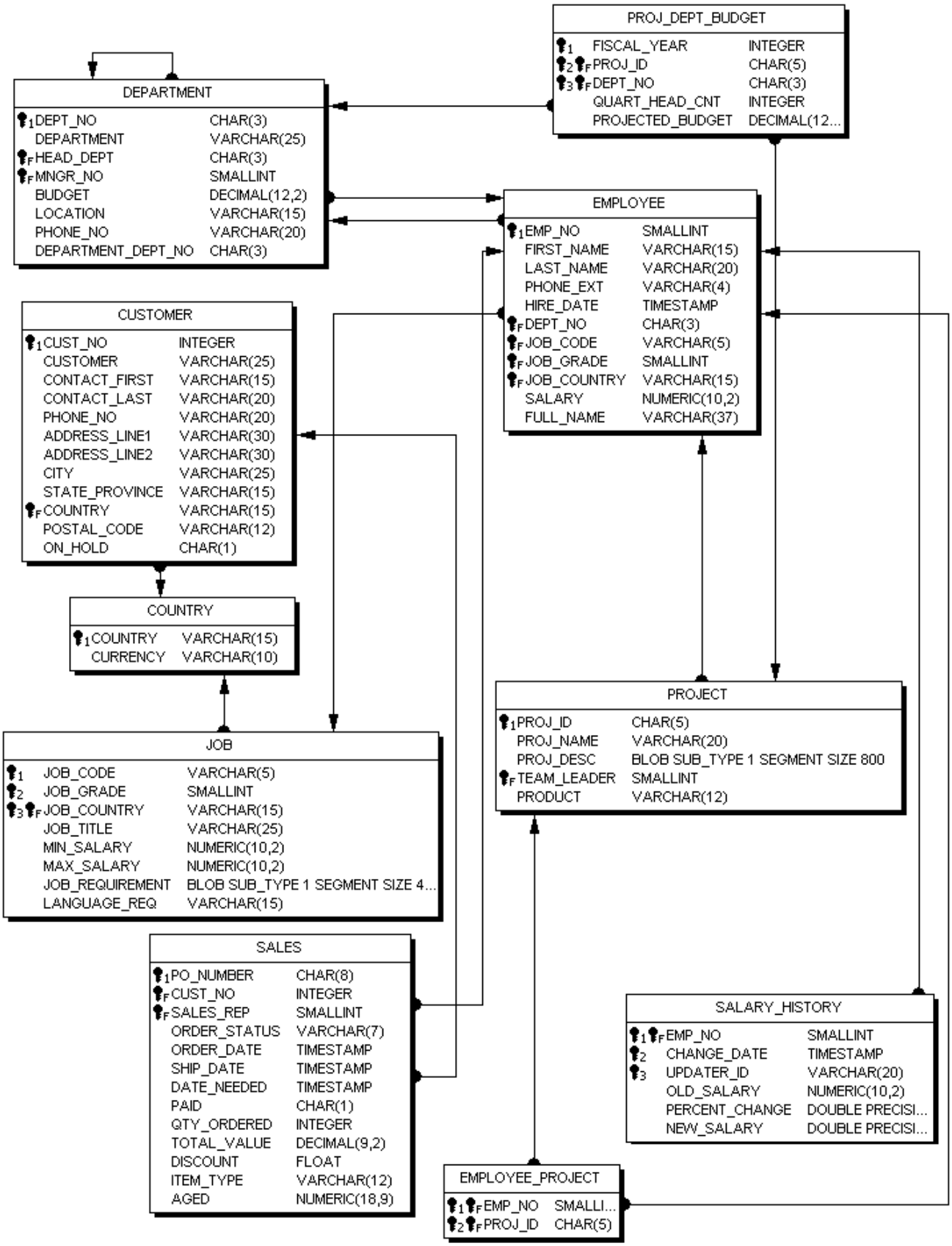

Рисунок 1. Схема базы данных employee.fdb