Министерство образования и науки Российской Федерации

Севастопольский государственный университет

ИЗУЧЕНИЕ ОСНОВ ЯЗЫКА МАНИПУЛИРОВАНИЯ ДАННЫМИ SQL НА БАЗЕ СЕРВЕРА FIREBIRD

Методические указания

к лабораторной работе №1

по дисциплине

“Управление данными”

для студентов специальности 09.03.02 –

"Информационные системы и технологии"

всех форм обучения

Севастополь
2014

УДК 004.92

Изучение основ языка манипулирования данными SQL на базе сервера Firebird.
Методические указания к лабораторной работе №1 по дисциплине “Управление данными”, для студентов всех форм обучения специальности 09.03.02 -"Информационные системы и технологии /Сост. Ю.В. Доронина, О.Л. Тимофеева, М.Р. Валентюк. - Севастополь: Изд-во СевНТУ, 2014.-28с.

Цель методических указаний: выработка у учащихся практических навыков по работе с реляционными базами данных.

Методические указания утверждены на заседании кафедры
Информационных Систем.
Протокол № от 2014 г.

Рецензент: доц. кафедры кибернетики и вычислительной техники, канд.техн.наук. Литвинова Л.А.

Допущено учебно-методическим центром в качестве методических указаний.

Содержание

Введение 4

Лабораторная работа № 1. Изучение архитектуры сервера Firebird. Язык SQL. Создание таблицы и элементарные выборки. 5

Библиографический список 17

Приложение А.Варианты заданий. 18

Приложение Б.SQL. 28

ВВЕДЕНИЕ

В связи с широким внедрением вычислительной техники во все области деятельности человека, включая и повседневную жизнь, с одной стороны, и все увеличивающиеся потоки информации, с другой стороны, в настоящее время большую популярность получили различные электронные справочники, информационно - поисковые системы. В большинстве случаев, такие системы по своим размерам являются малыми и средними базами данных.

При разработке таких систем применяется теория реляционных баз данных, использующая сложный математический аппарат. Проектированию реляционных БД посвящено множество монографий, учебников и статей.

Данные методические указания предназначены для формирования навыков работы с распределенными реляционными БД. Предложенный комплекс лабораторных работ включает базовые принципы и подходы, которые необходимы для организации работы с современными средствами хранения информации. Так, в лабораторной работе №1 рассматриваются правила организации работы в среде многопользовательской реляционной БД Firebird, а также основные операторы языка описания данных и языка манипулирования данными.

Лабораторные работы № 2,4,5,6 - базовые операции по манипулированию данными, отображение реляционных операций на языке манипулирования данными SQL, простые и сложные (вложенные) запросы на выборку данных. Лабораторная работа №3 - требования к проектированию набора отношений реляционной БД для обеспечения целостности (достоверности) данных. Лабораторная работа №7 – средства для автоматического формирования значения ключа вводимых данных и способы создания автоматически вызываемых процедур, обеспечивающих дополнительные условия поддержания целостности данных при выполнении операций, изменяющих хранимые данные. Лабораторная работа №8 – манипулирование базой данных с помощью операций реляционной алгебры и языка SQL.

ЛАБОРАТОРНАЯ РАБОТА № 1.
ИЗУЧЕНИЕ АРХИТЕКТУРЫ СЕРВЕРА FIREBIRD.
SQL. СОЗДАНИЕ ТАБЛИЦЫ И ЭЛЕМЕНТАРНЫЕ ВЫБОРКИ

1. Цель работы:

1.1. Изучить основы организации сервера Firebird; научится устанавливать соединение с сервером. Научиться создать базу данных и производить элементарные действия над ней.

1.2. Изучить формы оператора CREATE TABLE и простейшие формы оператора SELECT.

2. Основные положения

2.1. Описание сервера

Сервер Firebird является полнофункциональным сервером реляционных баз данных. Сервер выпускается под руководством несколько платформ (Windows, Linux, Solaris, freebsd, Darwin) в двух архитектурах – Super Server и Classic. Основное различие между ними состоит в том, что Classic создает параллельный процесс для каждого присоединяемого пользователя, а SuperServer состоит из одного процесса, который обрабатывает запросы клиентов в разных нитях (threads) этого же процесса. Архитектура Classic считается более надежной, а SuperServer более производительной.

Сервер на Windows запускается в качестве службы. Клиентские приложения могут присоединяться к нему несколькими способами: по протоколам NetBEUI, TCP-IP; локальное подключение (в случае, если вы работаете на машине, на которой запущен сервер). В дальнейшем рассматривается подключение к локальному серверу.

2.2. Создание и регистрация базы данных

Для создания базы данных необходимо запустить утилиту *isql*, которая находится в директории *bin* папки *Firebird* или через меню Пуск. Появится окно (рис. 1).

Рисунок 1 – Утилита *isql*

Оператор CREATE DATABASE создает новую базу данных (синтаксис оператора CREATE DATABASE приведен в Приложении Б):

```
SQL> create database 'c:\work\Ivanov.fdb'
CON> user 'SYSDBA' password 'masterkey'
CON> page_size = 4096
CON> default character set win1251;
```

Необходимо указать следующую информацию:

- имя файла (File Name) – имя физического файла базы данных, а также указать путь сохранения файла на диске;
- Имя пользователя (User Name) – SYSDBA (SYStem DataBase Administrator);
- Пароль (Password) –masterkey;
- Размер страницы базы данных задается равным 4096 (одно из допустимых значений);
- Обязательно надо установить набор символов по умолчанию (Default character set) в UTF8 или WIN1251.

Для соединения с уже существующей базой данных используется оператор CONNECT (синтаксис оператора CONNECT приведен в Приложении Б):

```
SQL> connect 'c:\work\Petrov.fdb' user 'SYSDBA' password 'masterkey';
```

После создания БД или соединения с уже существующей можно приступать к работе с ней.

2.3. Описание базовых операторов для работы с таблицами

2.3.1. CREATE TABLE

CREATE TABLE создает новую таблицу, ее столбцы и ограничения целостности в существующей базе данных. Пользователь, который создает таблицу, становится владельцем таблицы и получает полные привилегии на таблицу. Синтаксис оператора CREATE TABLE приведен в Приложении Б. В общем виде описание оператора можно представит следующим образом:

CREATE TABLE <имя таблицы> (<опред_столбца> [,<опред_столбца> | <ограничения> ...]);

Обязательно надо указать имя таблицы и определить как минимум один столбец.

CREATE TABLE поддерживает следующие возможности для определения столбцов:

Столбцы определяют имя и тип данных для данных, вводимых в столбец. Основанные на доменах столбцы наследуют все характеристики домена. На столбец может определить значение по умолчанию, атрибут NOT NULL, дополнительные ограничения CHECK или порядок сортировки.

Вычисляемые столбцы. Значение столбца вычисляется каждый раз при доступе к таблице. Если тип данных не определен, Firebird определяет тип данных результата операции. Столбцы, к которым обращается вычисление, должны быть определены раньше, чем вычисляемый столбец.

Описание типа данных для столбца типа *CHAR, VARCHAR* или *BLOB* может включать предложение *CHARACTER SET*, определяя специфическую кодировку для одиночного столбца. Иначе столбец использует определенную по умолчанию для базы данных кодировку. Если кодировка базы данных изменена, все столбцы впоследствии определенные имеют новую кодировку, но существующие столбцы не изменяются.

Примеры использования оператора CREATE TABLE

Рассмотрим таблицу «Служащий» (EMPLOYEE), имеющую следующую структуру.

Emp_Num – номер служащего, целочисленное, не может принимать значение NULL;
L_Name – фамилия, строка переменной длины с максимальной длиной 20 символов, не может принимать значение NULL;
F_Name – имя, строка переменной длины с максимальной длиной 10 символов, не может принимать значение NULL;
Dep_Num – номер отдела, целочисленное, не может принимать значение NULL;
Job_Cod – должность, строка переменной длины с максимальной длиной 10 символов;
Phone Ext – внутренний номер телефона, целочисленное, находится в переделах от 222 до 444;
Salary – оклад, значение от 0 до 9999,99.

Emp_Num	L_Name	F_Name	Dep_Num	Job_Cod	Phone Ext	Salary
111	Ivanov	Ivan	10	engineer	226	890,50
112	Petrov	Stepan	10	admin		1456,96
113	Sidorova	Irina	20	engineer	442	920,48

Рисунок 2 – Таблица «Служащий»

Следующая инструкция создает указанную таблицу:

```
CREATE TABLE EMPLOYEE
  (EMP_NUM INTEGER NOT NULL,
    L_NAME VARCHAR(20) NOT NULL,
    N_NAME VARCHAR(10) NOT NULL,
    DEP_NUM INTEGER NOT NULL,
    JOB_CODE VARCHAR(10),
    PHONE_EXT INTEGER CHECK (PHONE_EXT BETWEEN 222 AND 444),
    SALARY  DECIMAL(6,2) DEFAULT 0
);
```

Таблица «Отдел» (DEPARTMENT), представленная на рисунке 3, содержит следующие атрибуты:

Dep_Num – номер отдела, целочисленное, не может принимать значение NULL;
Dep_ Name – наименование отдела, строка переменной длины с максимальной длиной 20 символов;
Dep_ Head – номер служащего, руководителя отдела целочисленное;
Budget – бюджет, значение от 0 до 999999,99;
Location – адрес отдела, строка переменной длины с максимальной длиной 20 символов;
Phone_Num – номер телефона, целочисленное.

Dep_Num	Dep_Name	Dep_Head	Budget	Location	Phone_Num
10	Inspector		20000,00	fl 5, of 16-34	454601

20	construct	164	500000,0	fl 6-7	456701
30	security	46	100000,0	fl 10, of 1-22	442101

Рисунок 3 – Таблица «Отдел»

Следующая инструкция создает указанную таблицу:

```
CREATE TABLE DEPARTMENT
  (DEP_NUM INTEGER NOT NULL,
    DEP_NAME VARCHAR(20),
    DEP_HEAD INTEGER,
    BUDGET  DECIMAL(8,2) DEFAULT 0
    LOCATION VARCHAR(20),
    PHONE_NUM INTEGER
);
```

Таблица «Проект» (PROJECT), представленная на рисунке 4, содержит следующие атрибуты:
Pr_ID – код проекта, целочисленное, не может принимать значение NULL;
Pr_Name – наименование проекта, строка переменной длины с максимальной длиной 80 символов;
Pr_Descr – описание проекта,
Leader – номер служащего – руководителя проекта, целочисленное, не может принимать значение NULL;
Budget – бюджет проекта , значение от 0 до 999999,99.

PR_ID	PR_NAME	PR_DESCR	LEADER	BUDGET
1112	Data based …		113	20000,00
1114	Parallel algorithms …		112	50000,0
2222	Knowledge …		224	100000,0

Рисунок 4 – Таблица «Проект»

Следующая инструкция создает указанную таблицу:

```
CREATE TABLE PROJECT
 (PR_ID  INTEGER NOT NULL,
 PR_NAME VARCHAR(80),
 PR_DESCR VARCHAR(1000)
 LEADER INTEGER,
 BUDGET DECIMAL(8,2) DEFAULT 0
 );
```

Таблица «Проект-Служащий» (PROJECT- EMPLOYEE), представленная на рисунке 5, содержит следующие атрибуты:
Emp_Num – номер служащего, целочисленное, не может принимать значение NULL;
Pr_ID – код проекта, целочисленное, не может принимать значение NULL;
Data_S – содержит дату начала работ;
Data_F – содержит дату сдачи проекта.

PR_ID	EMP_NUM	DATA_S	DATA_F
1112	111	25.08.2004	2.04.2005
1112	112	12.06.2004	22.05.2005
2222	111	22.10.2004	22.01.2005
2222	113	2.08.2004	22.08.2005

Рисунок 5 – Таблица «Проект-Служащий»

Следующая инструкция создает указанную таблицу:

```
CREATE TABLE PROJECT_ EMPLOYEE
  ( PR_ID  INTEGER NOT NULL,
    EMP_NUM INTEGER NOT NULL,
    DATA_S   DATE
    DATA_F   DATE
```

);

2.3.2. INSERT

INSERT добавляет одну или более новых строк данных к существующей таблице. Синтаксис оператора INSERT приведен в Приложении Б. Значения вставляются в столбцы строки по порядку их следования, если не задан факультативный список столбцов. Если список столбцов задан на множестве всех доступных столбцов, то во все не перечисленные столбцы автоматически вставляется или значение по умолчанию, или NULL.

Если факультативный список столбцов упущен, предложение VALUES должно содержать значения для всех столбцов таблицы.

Чтобы вставить одну строку данных, должно присутствовать предложение VALUES и содержать определенный список значений.

Чтобы вставить несколько, строк данных, определите <*select_expr*>, которое возвращает уже существующие данные из другой таблицы. Выбранные строки должны соответствовать списку столбцов.

Предупреждение: Допустимо выбирать из той же таблицы, в которую строки вставляются, но эта практика не рекомендуется, так как подобные действия могут привести к бесконечным вставкам строк (зацикливанию).

Примеры использования оператора *INSERT*

1. Следующая инструкция добавляет строку в таблицу «Отдел»

```
INSERT INTO  DEPARTMENT VALUES (20, 'construct', 164, , 'fl  6-7',456701);
```

2. Следующая инструкция добавляет строку в таблицу «Служащий», присваивает значения шести столбцам:

```
INSERT INTO EMPLOYEE (EMP_NUM, L_NAME, N_NAME, DEP_NUM, SALARY, JOB_CODE)
                      VALUES ( 112, 'Petrov', 'Stepan',  10, 1456.96,  'admin');
```

3. Следующая инструкция определяет значения, чтобы вставить в таблицу, используя инструкцию SELECT:

```
INSERT INTO PROJECTS
  SELECT * FROM NEW_PROJECTS
    WHERE NEW_PROJECTS.START_DATE > '6-JUN-1994';
```

2.3.3. SELECT

SELECT возвращает данные из таблицы, представления или сохраненной процедуры. Синтаксис оператора SELECT приведен в приложении В. Различные инструкции SELECT выполняют следующие действия:

- Возвращают одиночную стоку или часть строки из таблицы. Это операция упоминается, как *sigleton выбор*.
- Непосредственно возвращают список строк или список частичных строк из таблицы.
- Возвращают связанные строки, или частичные строки из *join* двух или более таблиц.
- Возвращают все строки, или частичные строки из *union* двух или более таблиц.

Любая инструкция SELECT содержит два обязательных предложения (SELECT, FROM) и возможно другие предложения (WHERE, GROUP BY, HAVING, UNION, PLAN, ORDER BY). Предложения SELECT и FROM обязательны и для sigleton, и для multi-row SELECT; все другие предложения перечисленные ниже факультативны. Назначение каждого предложения оператора SELECT приведено в таблице 5.

Таблица 1 - Назначение предложений, входящих в оператор SELECT

Предложение	Назначение
SELECT	Список столбцов, которые возвращаются.
FROM	Определяет таблицы, в которых ищутся значения.
WHERE	Определенное условие поиска, которое используется, чтобы выбрать необходимые стоки из множества всех строк. Предложение WHERE может содержать инструкцию SELECT, которая упоминается, как подзапрос
GROUP BY	Группирует возвращенные строки, основываясь на общих значениях столбцов. Используется совместно с HAVING.
HAVING	Определяет условие поиска для группы записей. Строки отбираются из значений столбцов, указанных в GROUP BY и значений агрегатных функций, вычисленных для

	каждой группы, образованной GROUP BY.
UNION	Комбинирует результаты двух или более инструкций SELECT, создавая одиночную динамическую таблицу, исключая повторяющиеся строки.
ORDER BY	Определяет порядок сортировки строк возвращенных SELECT, по умолчанию в возрастающем порядке (ASC), или в убывающем порядке (DESC).
PLAN	Определяет план запроса, который будет использоваться оптимизатором запроса вместо обычного выбора.

Примеры использования оператора *SELECT*

1. Следующая инструкция выбирает столбцы из таблицы «Отдел»:

```
SELECT  DEP_NAME, BUDGET, PHONE_NUM  FROM DEPARTMENT;
```

2. Следующая инструкция использует шаблон *, что бы выбрать все столбцы и строки из таблицы «Проект»:

```
SELECT * FROM PROJECT ;
```

3. Следующая инструкция использует агрегатную функцию, что бы сосчитать все строки в таблице, которые удовлетворяют условиям поиска определенным в предложении WHERE, количество проектов с бюджетом выше 50000:

```
SELECT COUNT (*) FROM PROJECT WHERE BUDGET > 50000;
```

4. Следующая инструкция использует составное условие поиска, определенное в предложении WHERE, позволяет определить фамилии и должности сотрудников 10 отдела, имеющих оклад выше 600:

```
SELECT L_NAME, STATE FROM EMPLOYEE  WHERE DEP_NUM=10 AND SALARY>600;
```

5. Следующая инструкция выбирает два столбца и сортирует возвращенные строки по второму столбцу:

```
SELECT L_NAME, STATE FROM EMPLOYEE
 ORDER BY STATE;
```

6. Следующая инструкция позволяет оценить в каких проектах служащий принимает участие и вывести информацию с учетом даты окончания служащим работ по проекту

```
SELECT PR_NAME, EMP_NUM, DATA_F FROM PROJECT_EMPLOYEE
   WHERE EMP_NUM=112
    ORDER BY DATA_F ;
```

7. Следующая инструкция использует в условии поиска конструкцию IN (входит), позволяет определить наименования проектов над которыми работают служащие номера, которых перечислены в списке:

```
SELECT PR_NAME  FROM PROJECT_EMPLOYEE
   WHERE  EMP_NUM  IN (111, 113);
```

3. Ход работы

1. Ознакомиться с описанием организации работы реляционных БД.
2. Создать тестовую БД, в соответствии с п.2.2.
3. В соответствии, с п.2.3:
   - создать тестовую таблицу,

- занести в таблицу пять кортежей,
- просмотреть содержимое таблицы.

5. Определить номер своего варианта. Номер варианта(x) определяется как x = N mod 20, где N – номер студента в группе.
6. По номеру своего варианта определить имя таблицы, которую надо будет создать при выполнении работы (Приложение А).
7. Ознакомиться с описанием базовых операторов для работы с таблицами.
8. Создать таблицу. Особое внимание надо уделить описанию первичного ключа, значений по умолчанию, описателям NOT NULL и конструкции CHECK.
9. Занести в таблицу образцы данных оператором INSERT INTO. Необходимо занести не менее 10 строк. Внимание! После того, как в таблицу занесены образцы данных, менять структуру таблицы можно только оператором ALTER TABLE.
10. Создать запрос, выводящий все строки таблицы.
11. Создать запрос, задающий порядок столбцов, отличный от исходного.
12. Продемонстрировать действие модификатора DISTINCT.
13. Ограничить вывод запроса, используя WHERE с простым условием.
14. Ограничить вывод запроса, используя WHERE и составное условие.
15. Продемонстрировать действие специальных функций IN, BETWEEN, LIKE, и IS NULL в условии.
16. Продемонстрировать работу специальных функций с условием NOT.

4. Содержание отчета

1. Отчет состоит из титульного листа, цели работы, описания процесса выполнения работы и вывода.
2. Отчет должен содержать описание действий студента по конкретному варианту.
3. Обязательно должны быть указаны фактические значения, вводимые в диалоговые окна.
4. . Отчет должен содержать:
   - таблицу исходных данных,
   - тексты запросов,
   - результаты их выполнения.

5. Контрольные вопросы

1. Объясните отличие архитектуры superserver и classic?
2. Какая информация указывается при создании базы данных?
3. Базовое понятие многопользовательских систем «транзакция»?
4. Модели данных. Реляционная модель данных?
5. СУБД – назначение?
6. Основные понятие реляционных БД: отношение, атрибут, кортеж, домен, схема отношения?
7. Правила задания таблиц?
8. Ключевой атрибут, первичный ключ?
9. Требования к вводу данных оператором INSERT INTO?

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Боуман Д.С. Практическое руководство по SQL. Использование диалектов SQL./Д.С.Боуман. –М.;СПб.;К.: Вильямс, 2001.- 351 с.
2. Боуман Д.С. Практическое руководство по SQL. Использование языка структурированных запросов./ Д.С.Боуман –М.; СПб.;К.: Вильямс, 2001.- 334 с.
3. Елланд Ф.Д. Основі концепции баз данніх./Ф.Д. Елланд. –М.; СПб.;К.: Вильямс, 2002.- 235 с.
4. Конноли Т. Базы данных. Проектирование, реализация и сопровождение теория и практика./Т.Конноли. –М.; СПб.;К.: Вильямс, 2001.- 1111 с.
5. Кузнецов С. SQL. Язык реляционных Баз данных./С. Кузнецов. –М.: Майор, 2001. -191 с.
6. Ульман Дж.Д Введение в системы баз данных./Дж.Д. Ульман.–М.:Лори, 2000. -374 с.

ПРИЛОЖЕНИЕ А. Варианты заданий

(Обязательное)

Варианты

Вариант 1

На рисунке А.1 изображена структура системы, которая содержит информацию о публикациях по ряду выбранных тем.

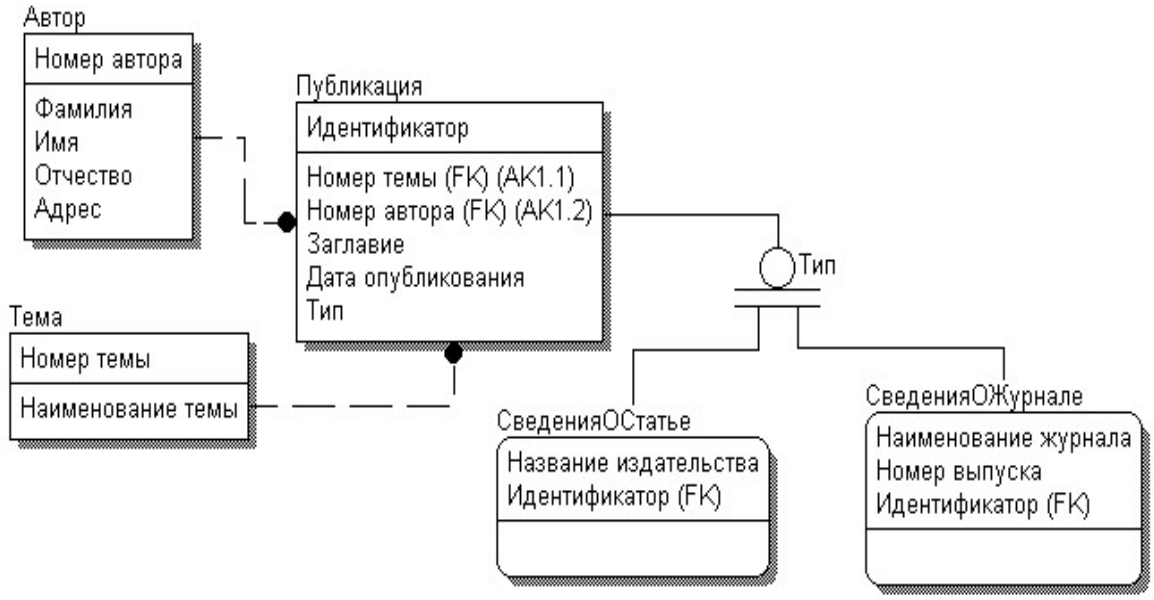


Рисунок А.1.

Вариант 2

На рисунке А. 2 изображена структура системы, которая содержит информацию о внутренней системе обучения в большой промышленной компании.

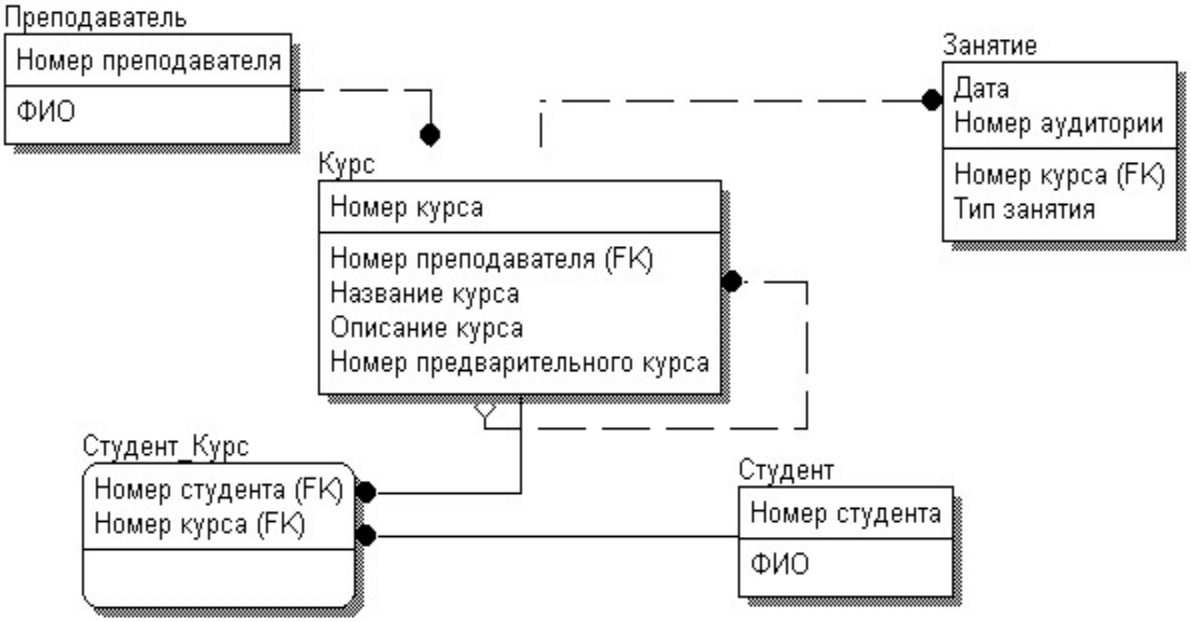


Рисунок А.2.

Вариант 3

На рисунке А.3 изображена структура системы, которая содержит информацию о пациентах клиники.

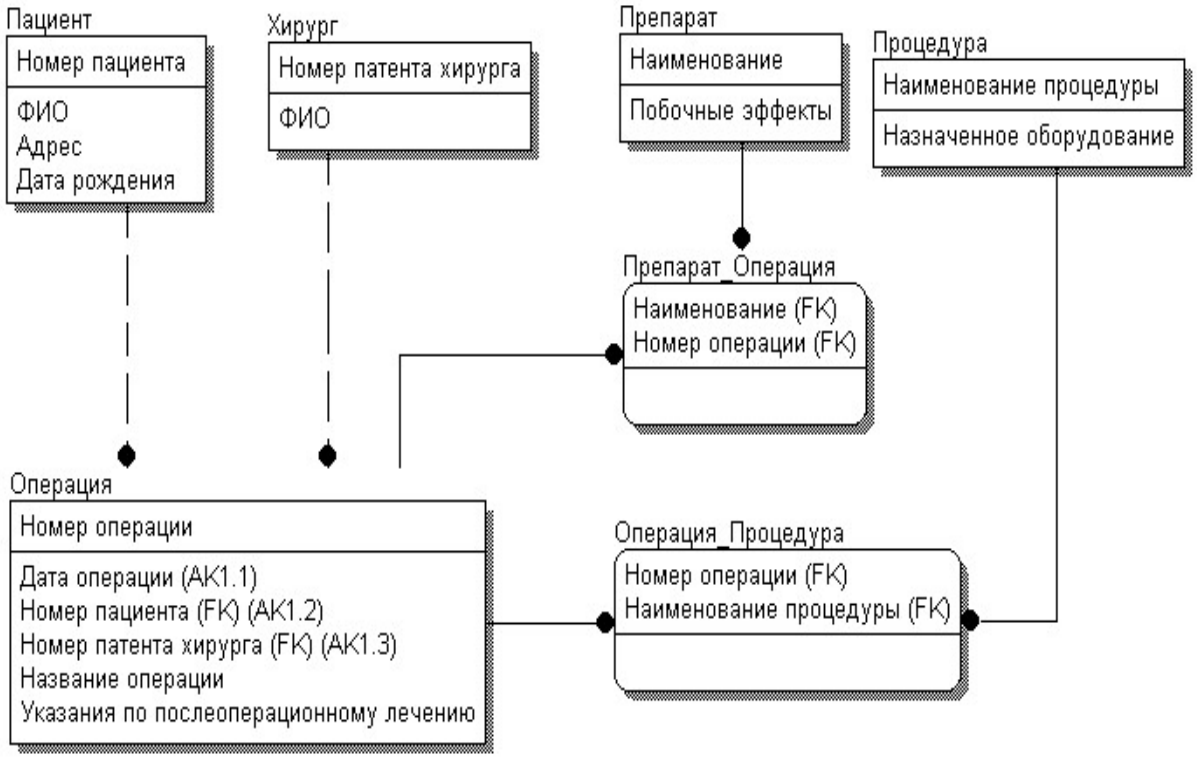


Рисунок А.3.

Вариант 4

На рисунке А.4 изображена структура системы, которая содержит анкетные данные служащих конторы.

Типы отношений: состоят в браке, является родителем

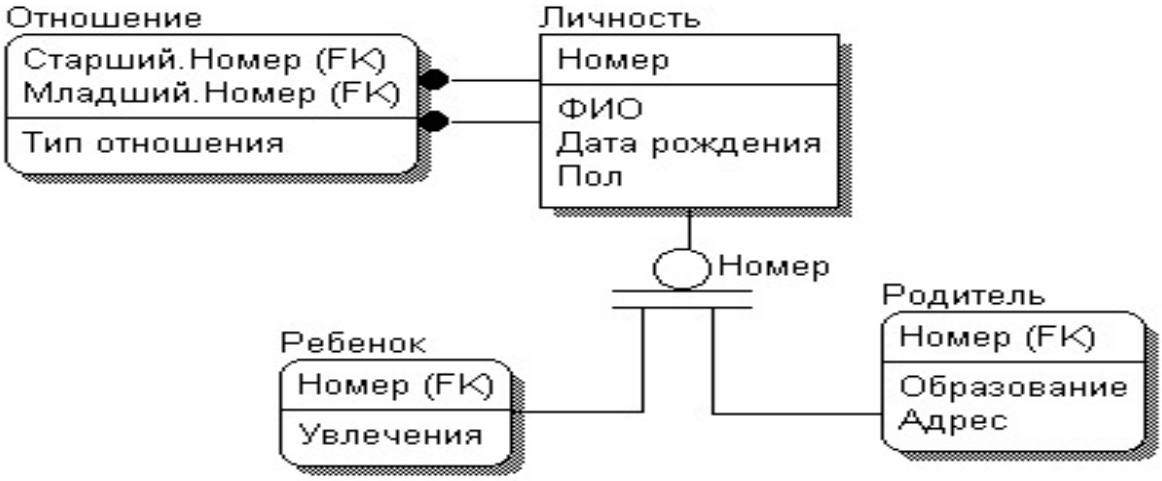


Рисунок А.4.

Вариант 5

На рисунке А.5 изображена структура системы, которая содержит информацию о торговых фирмах.

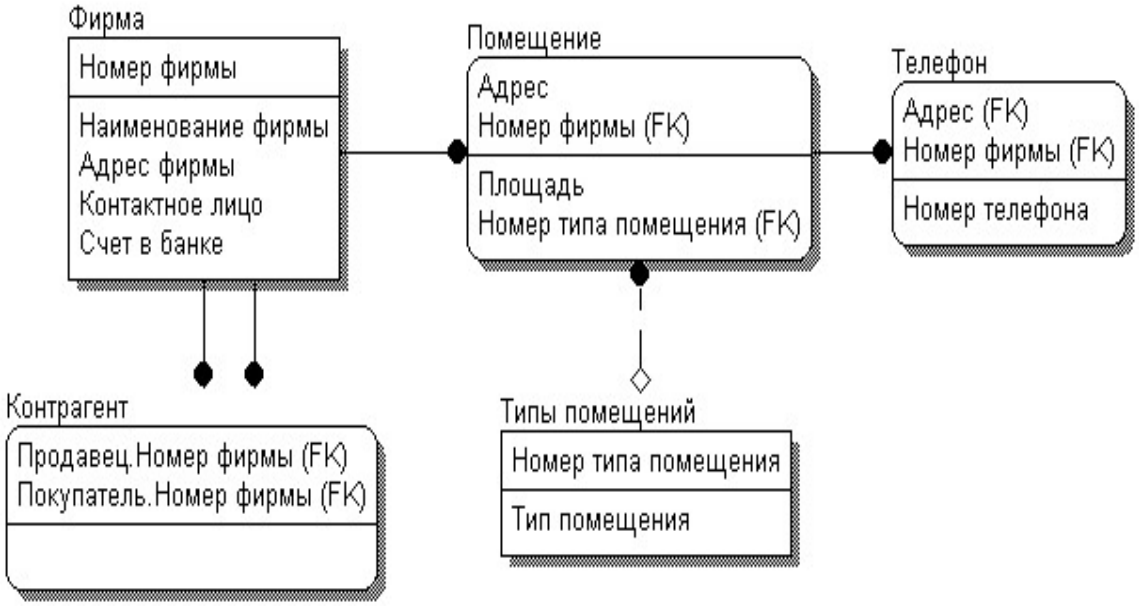


Рисунок А.5.

Вариант 6

На рисунке А.6 изображена структура системы, которая содержит каталог работ.

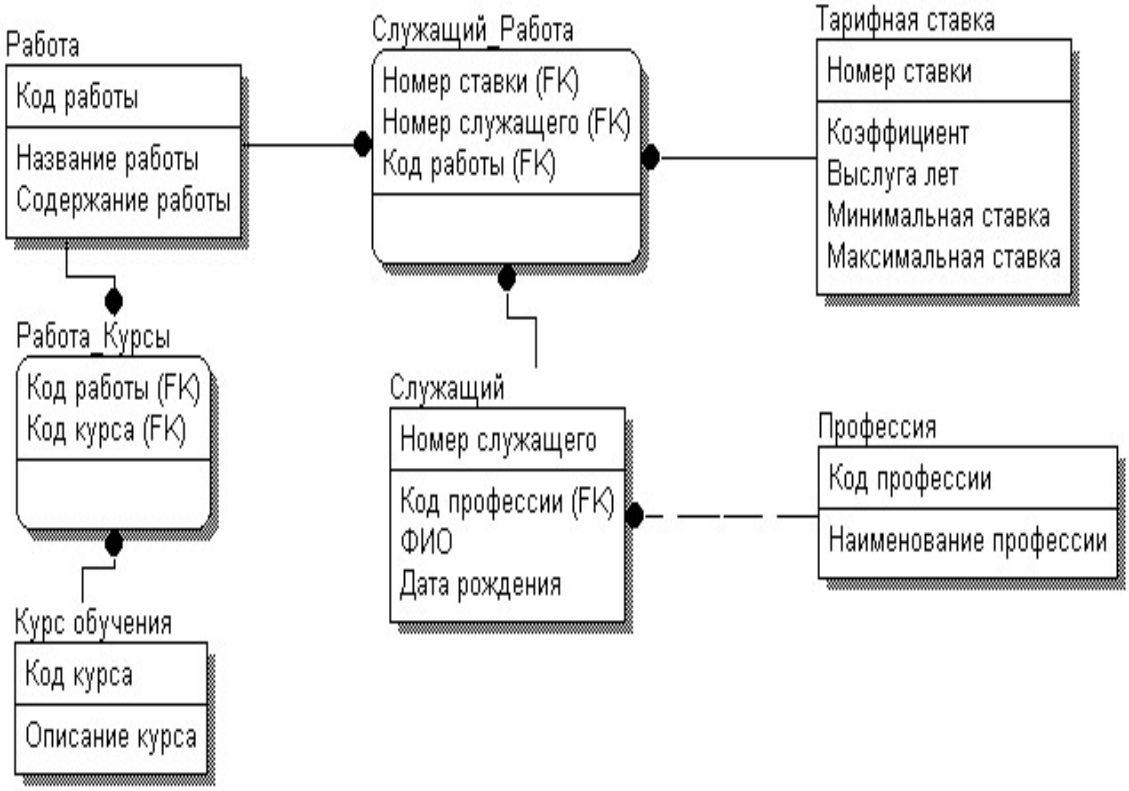
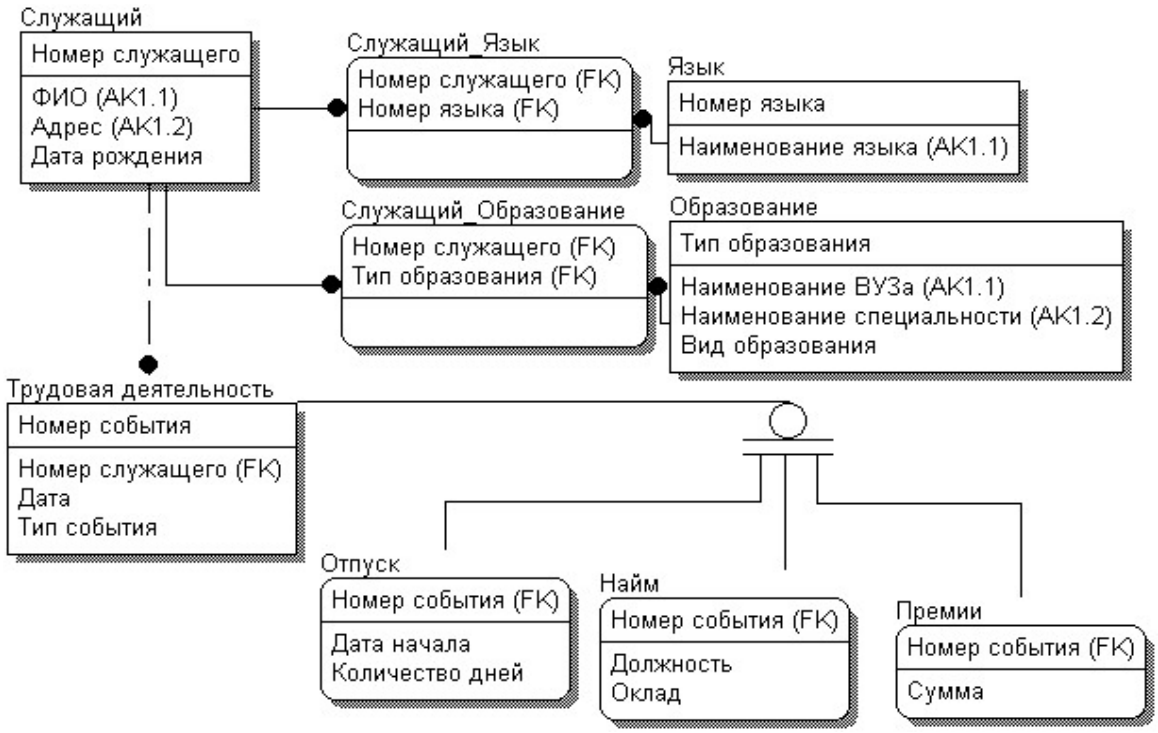


Рисунок А.6.

Вариант 7

На рисунке А.7 изображена структура системы, которая содержит информацию о служащих.



Вид образования - высшее, среднее, неполное высшее. Тип события – отпуск, премирование, события связанные с наймом (в том числе изменение в должности и окладе).

Рисунок А.7.

Вариант 8

На рисунке А.8 изображена структура системы, которая содержит информацию о факультетах вида:

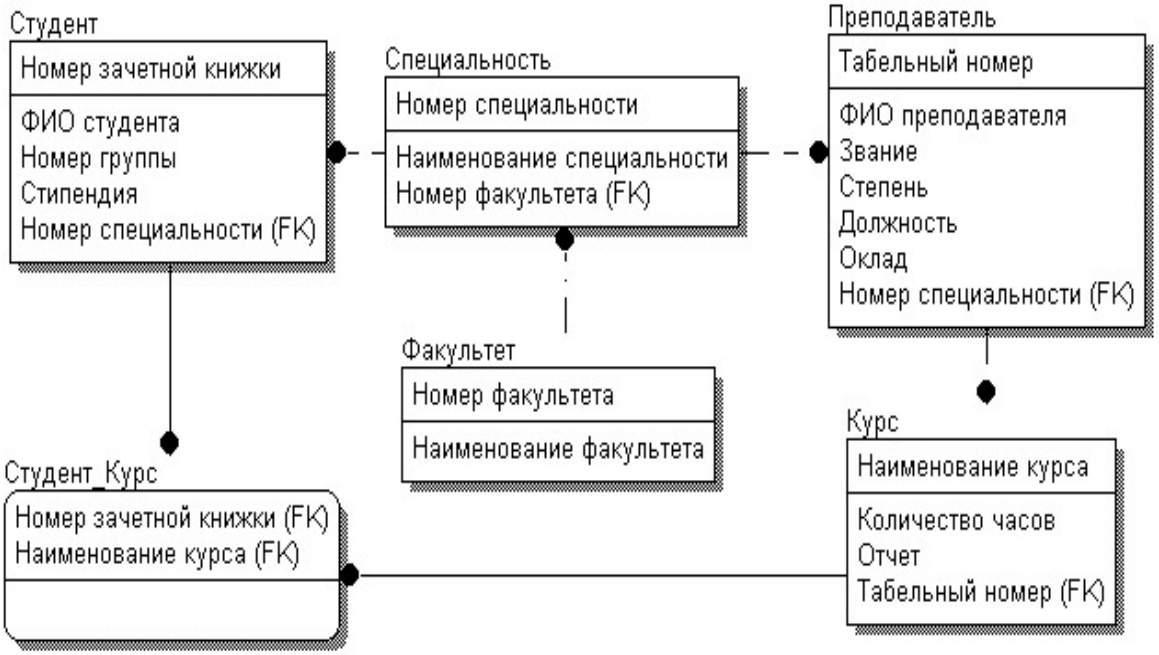


Рисунок А.8.

Вариант 9

На рисунке А.9 изображена структура системы, которая содержит информацию об учебных заведениях, абитуриентах и экзаменах

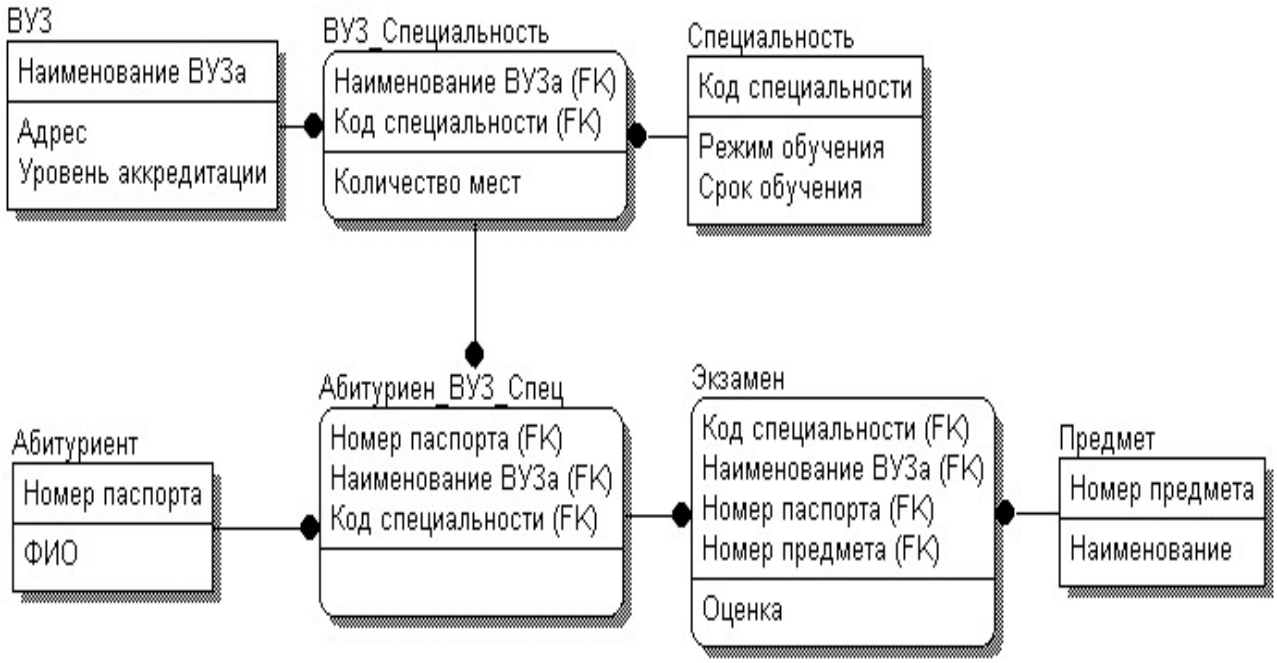
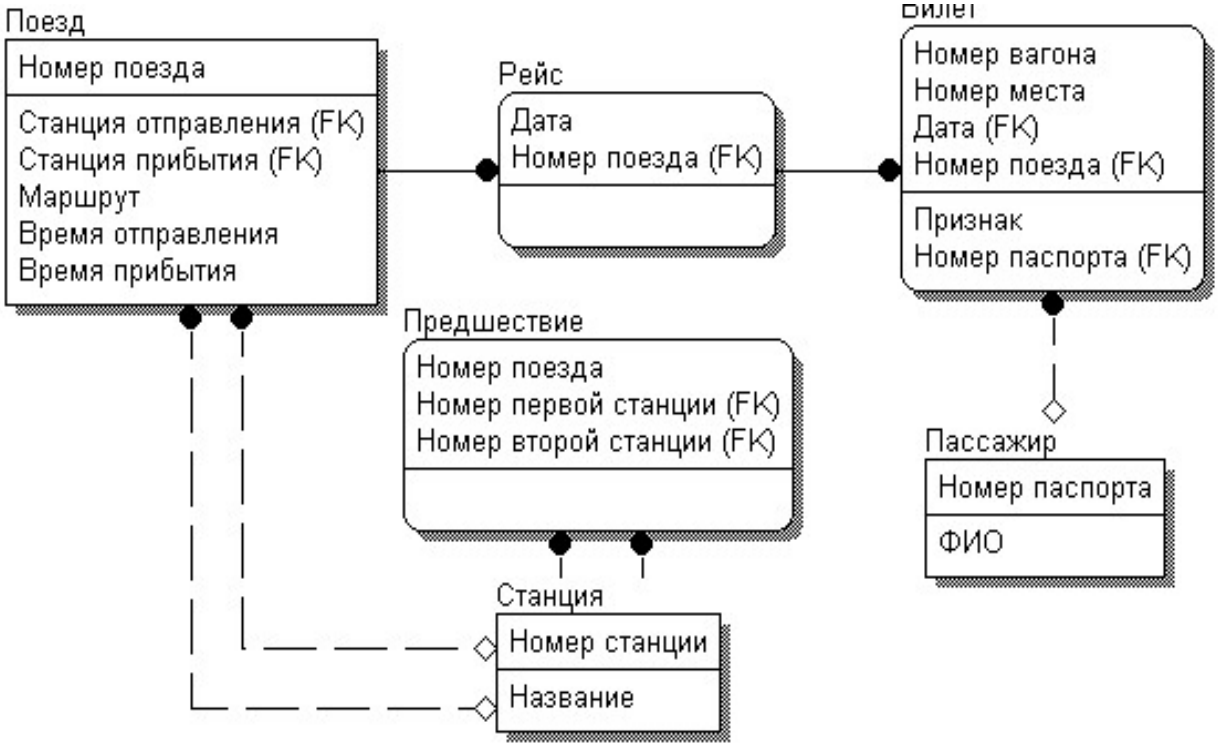


Рисунок А.9.

Вариант 10

На рисунке А.10 изображена структура системы, которая содержит информацию о движении поездов.



Признак билета – забронирован или выкуплен.

Рисунок А.10.

Вариант 11

На рисунке А.11 задана структура системы, которая содержит информацию о результатах экзаменов по каждой кафедре:

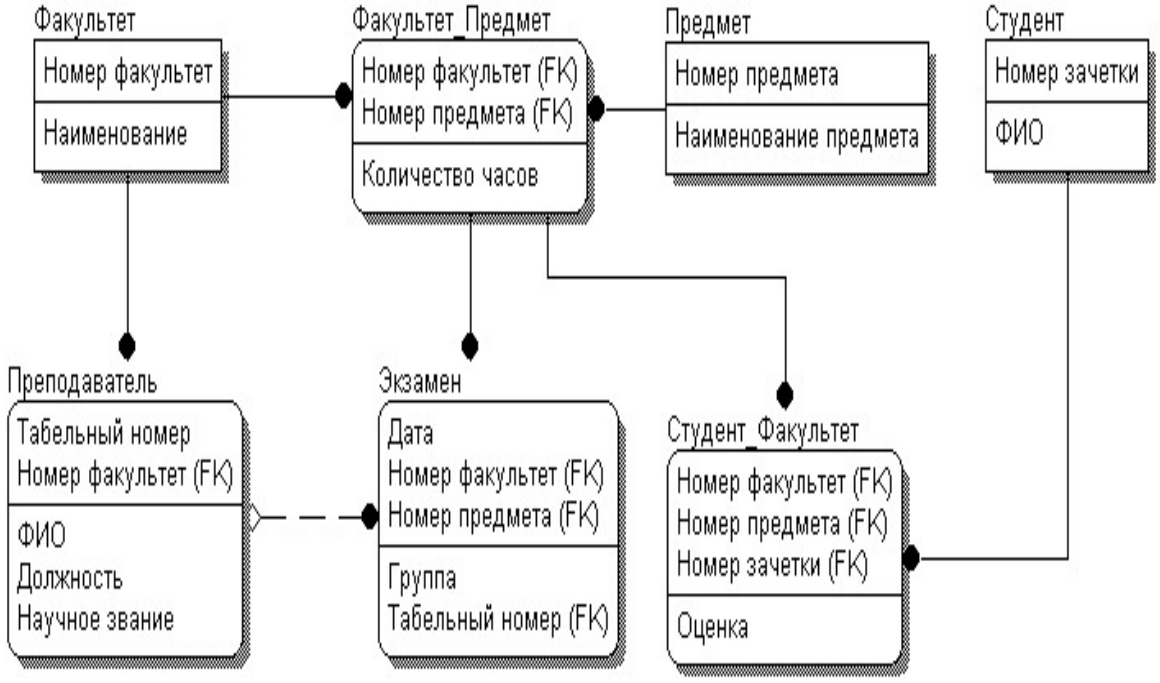


Рисунок А.11.

Вариант 12

На рисунке А.12. изображена структура системы, которая содержит картотеку осужденных лиц:

Рисунок А.12.

Вариант 13

На рисунке А.13 изображена структура системы, которая содержит информацию о футбольных командах:

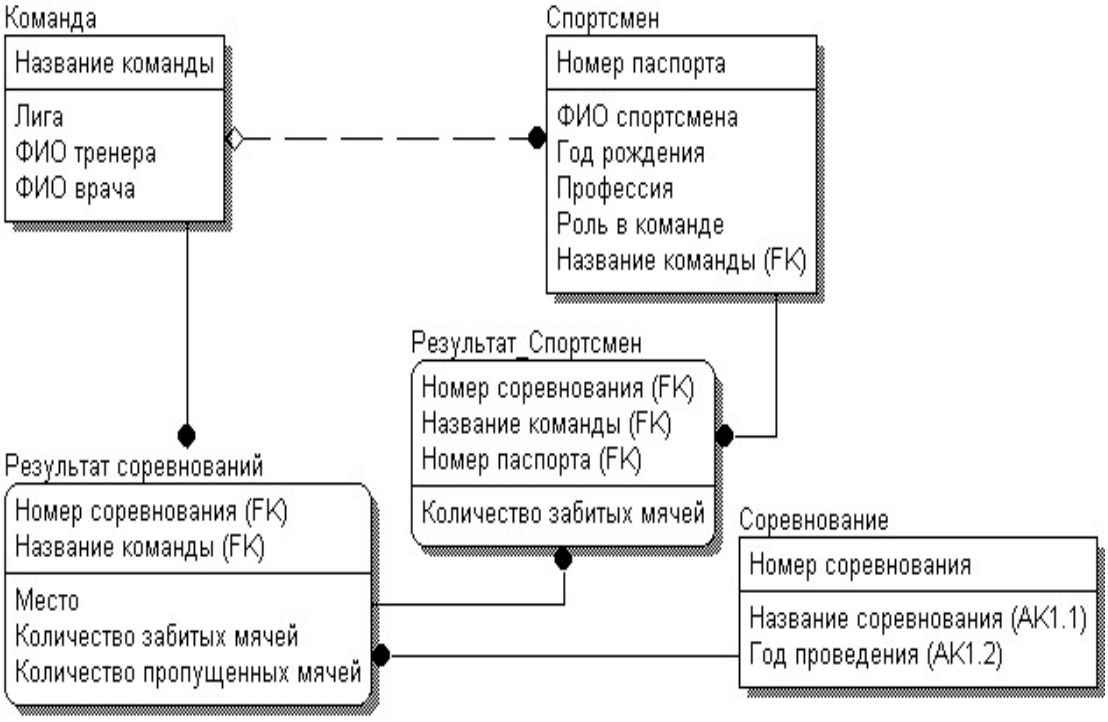
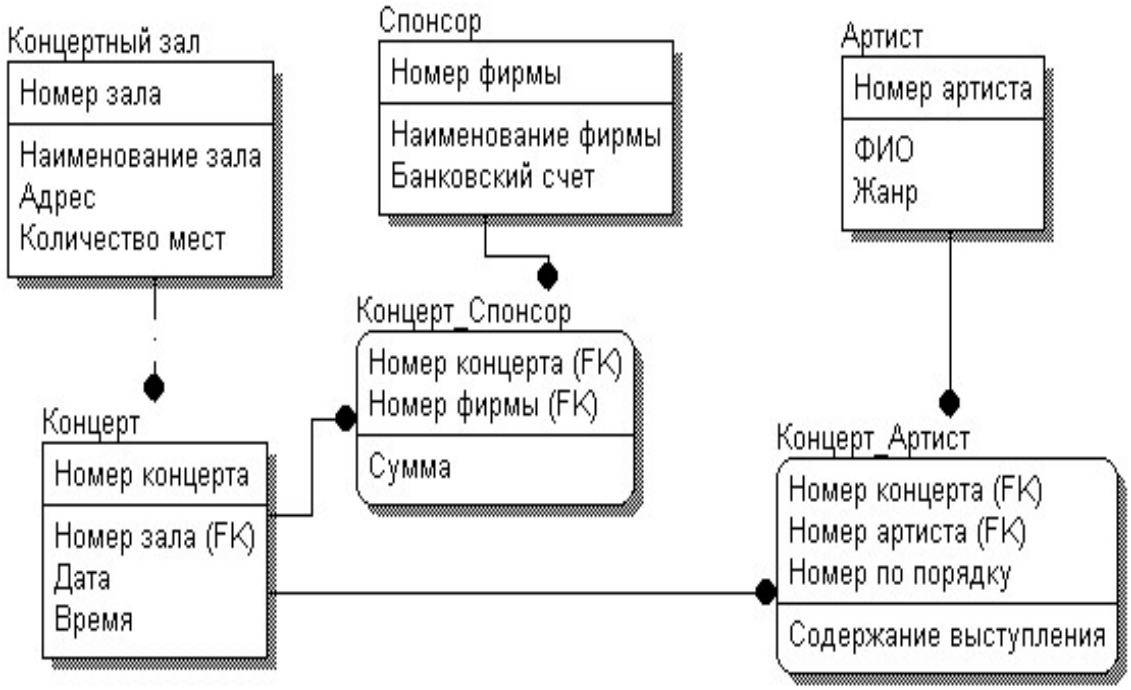


Рисунок А.13.

Вариант 14

На рисунке А. 14 изображена структура системы, которая содержит информацию о концертах:



Предусмотреть жанр «конферансье» - ведущий концерта.

Рисунок А.14.

Вариант 15

На рисунке А.15 изображена структура системы, которая содержит картотеку фирм:

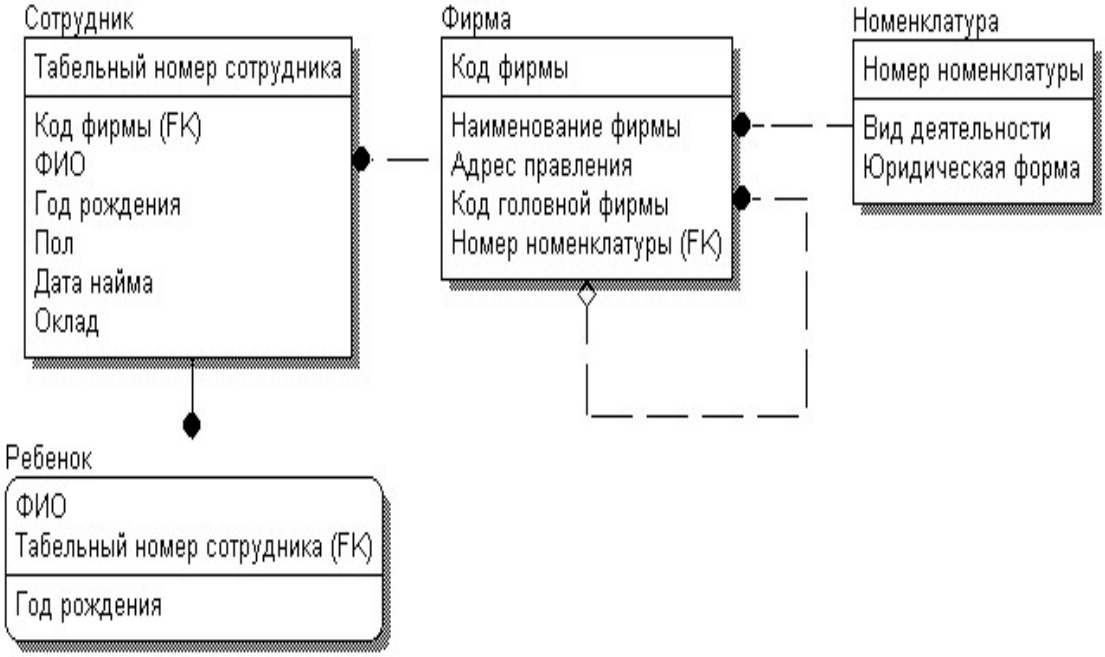
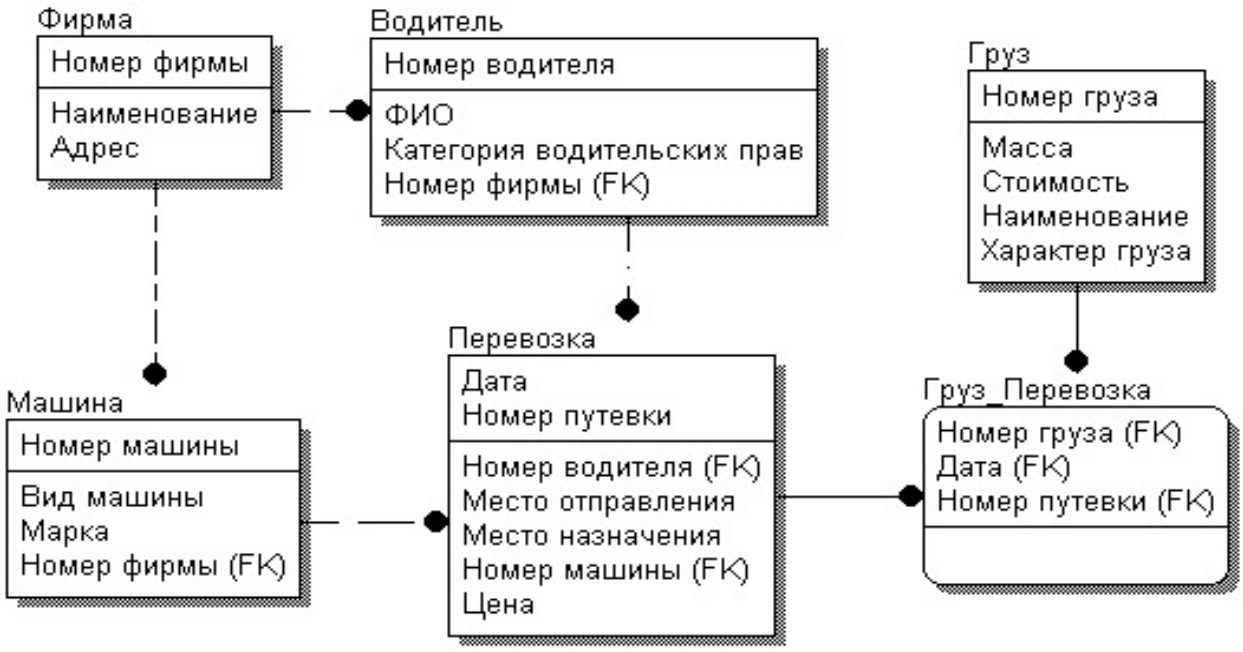


Рисунок А.15.

Вариант 16

На рисунке А.16 изображена структура системы, которая содержит информацию о грузовых перевозках, осуществляемых различными фирмами.



Категория водительских прав –A, B, C
Характер груза - твердый, жидкий и т.д.

Рисунок А.16.

Вариант 17

На рисунке А. 17 изображена структура системы, которая содержит информацию для бронирования мест в гостинице для командированных:

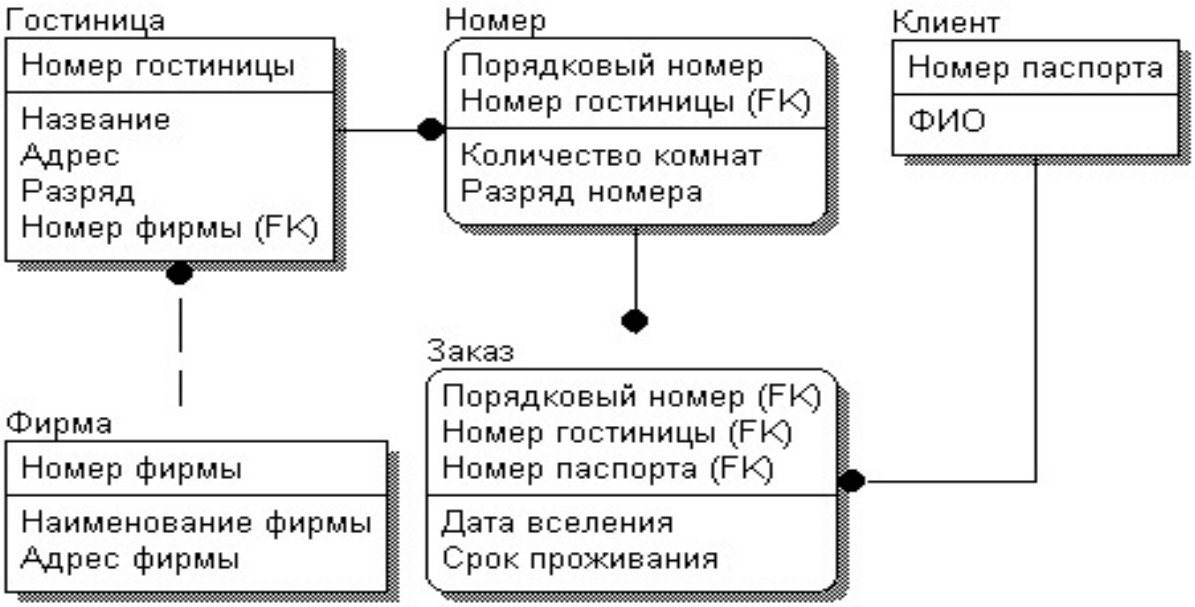


Рисунок А.17.

Вариант 18

На рисунке А.18 изображена структура системы, которая содержит информацию о лечении различных заболеваний:

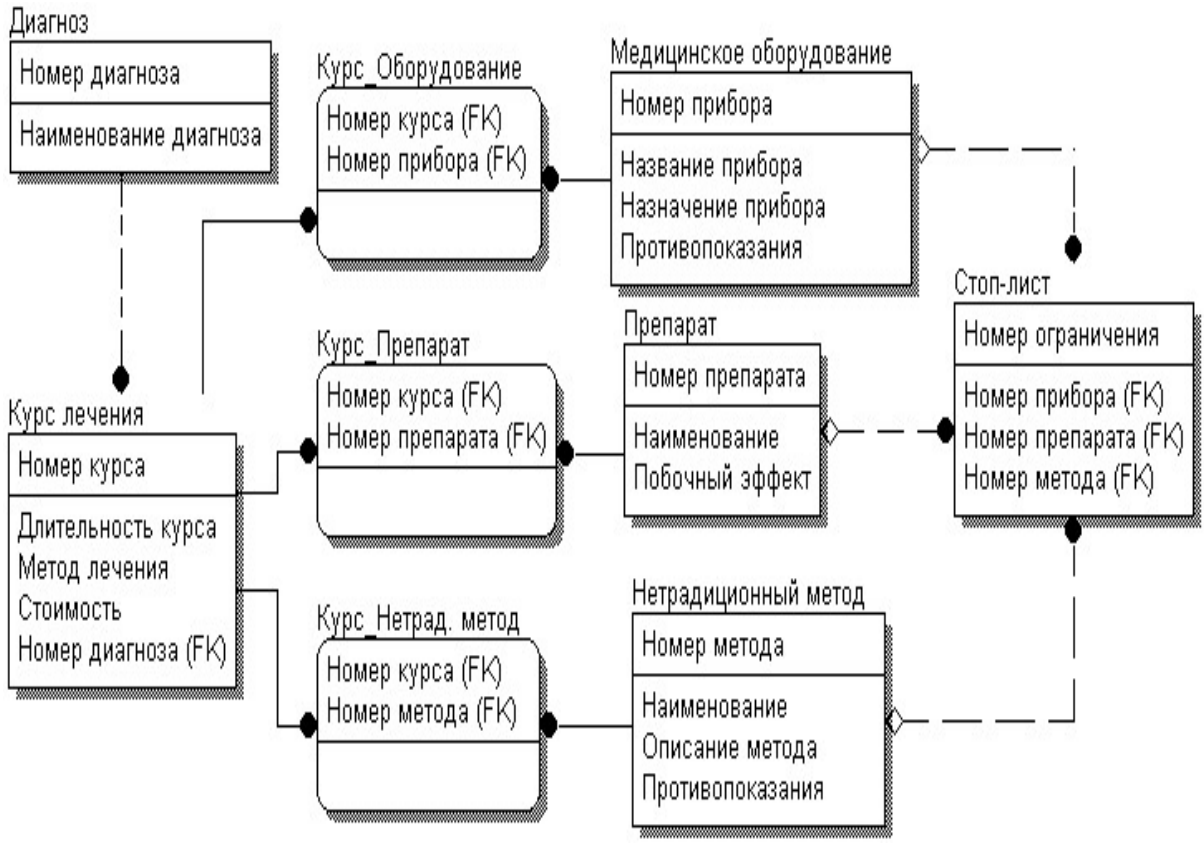


Рисунок А.18.

Вариант 19

На рисунке А. 19 изображена структура системы, которая содержит информацию о вычислительных центрах города:

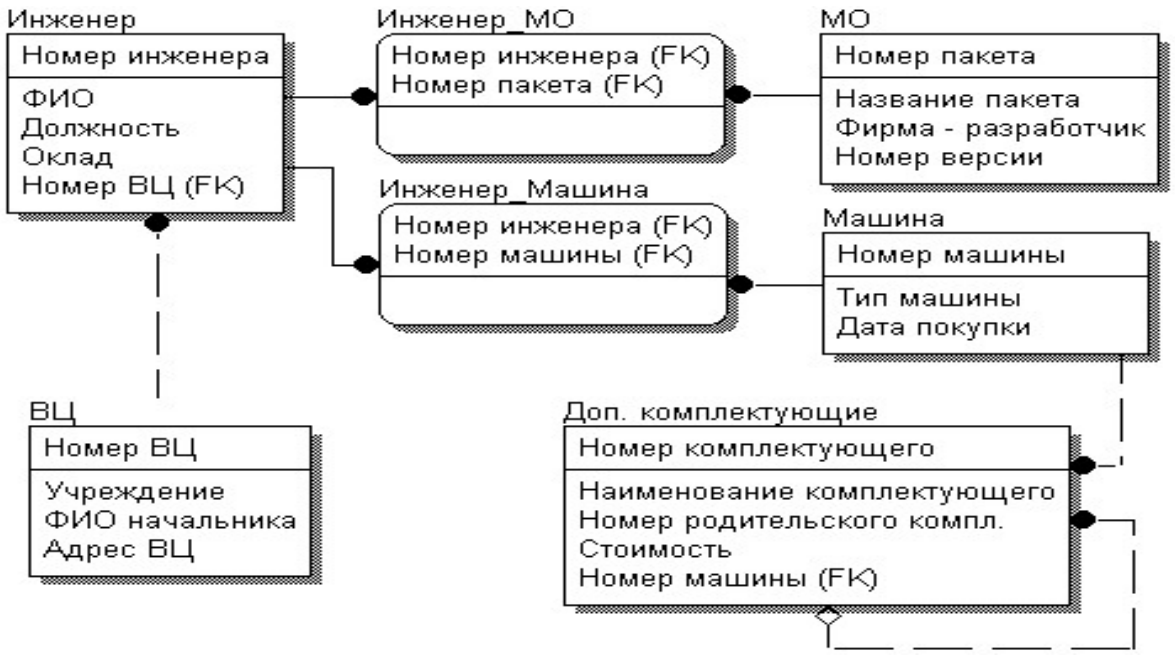


Рисунок А.19.

Вариант 20

На рисунке А.20 изображена структура системы, которая содержит информацию о политических партиях:

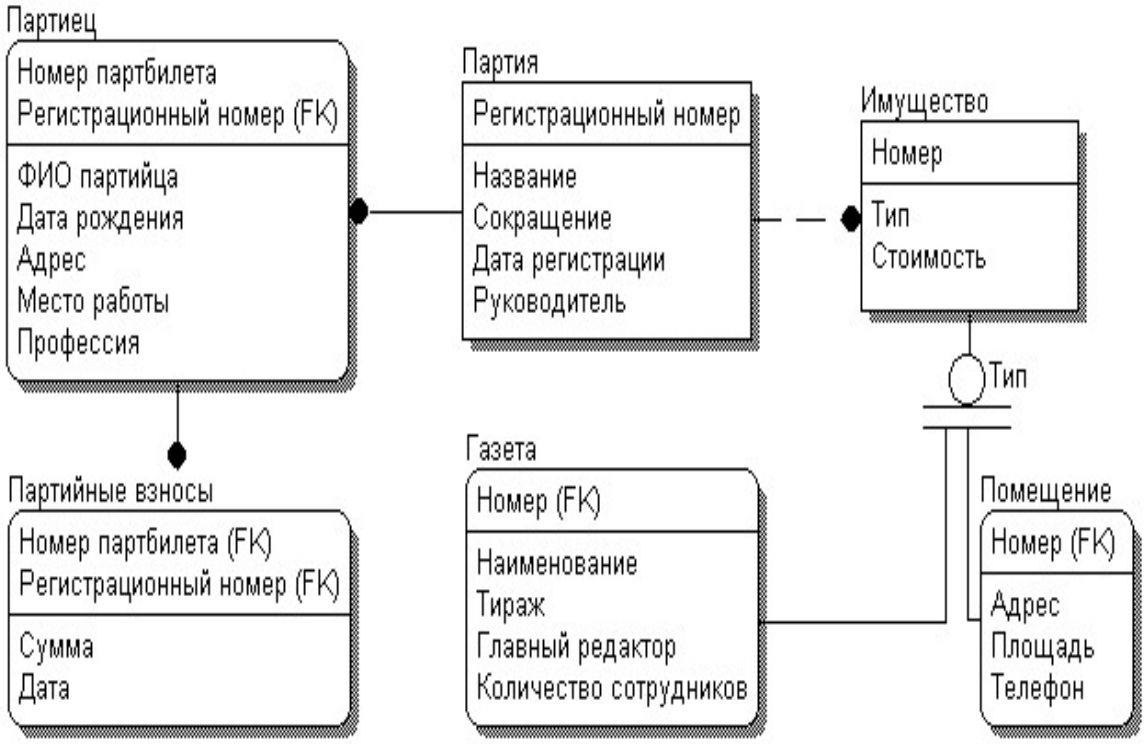


Рисунок А.20.

ПРИЛОЖЕНИЕ Б
(обязательное)
SQL.

Синтаксис оператора CREATE DATABASE

```
CREATE {DATABASE | SCHEMA} '<спецификация файла>'
[USER '<имя пользователя>' [PASSWORD '<пароль>']]
[PAGE_SIZE [=] <целое>]
[LENGTH [=] <целое> [PAGE[S]]]
[DEFAULT CHARACTER SET <набор символов>]
[<вторичный файл>]...;

<вторичный файл> ::= FILE '<спецификация файла>'
[LENGTH [=] <целое> [PAGE[S]]]
[STARTING [AT [PAGE]] <целое>]
```

Таблица Б.1. – Описание конструкций оператора CREATE DATABASE

Аргумент	Описание
'<спецификация файла>'	Спецификация файла задает полный путь к создаваемому файлу базы данных и имя файла, включая его расширение. Сам файл должен отсутствовать на внешнем носителе.
USER '<имя пользователя>'	Имя пользователя-владельца базы данных. Может содержать до 31 символа. Нечувствительно к регистру.
PASSWORD '<пароль>'	Пароль пользователя-владельца базы данных. Может содержать до 64 символов, однако только первые 8 имеют значение. Чувствителен к регистру.
PAGE_SIZE	Задает размер страницы базы данных. Допустимы значения 4096 (значение по умолчанию), 8192 и 16384.Если вы зададите неправильное значение размера страницы, то система не выдаст сообщения об ошибке, а установит размер до ближайшего меньшего числа. Если указать значение меньше чем 4096, то будет выбрано значение по умолчанию — 4096.
LENGTH	Задает количество страниц в первичном файле базы данных.
DEFAULT CHARACTER SET	Задает набор символов по умолчанию для строковых типов данных в базе данных.
STARTING AT PAGE	Страница, с которой начинается вторичный файл базы данных.

Синтаксис оператора CONNECT

```
CONNECT '<спецификация файла>'
  [USER '<имя пользователя>' [PASSWORD '<пароль>']]
  [CACHE <целое> [BUFFERS]]
  [ROLE '<имя роли>'];
```

В операторе указывается имя первичного файла базы данных. Имя пользователя (предложение USER) и его пароль (PASSWORD) должны задавать пользователя, описанного в системе. Имя пользователя может содержать до 31 символа. Оно нечувствительно к регистру. Пароль чувствителен к регистру. Может содержать до 32 символов, однако только первые восемь имеют значение.

Предложение CACHE задает количество буферов кэш-памяти для соединения. По умолчанию 256. Перекрывает значение, указанное в конфиге или базе данных, только если больше указанных там значений.

Предложение ROLE задает имя роли, с которой пользователь соединяется с данной базой данных. Имя роли может содержать до 31 символа

Синтаксис оператора CREATE TABLE

```
CREATE TABLE <имя таблицы>
[EXTERNAL [FILE] '<спецификация файла>']
(<определение столбца>
[, <определение столбца> | <ограничение таблицы>]...);

<определение столбца> ::= <имя столбца>
{ <тип данных>
| <имя домена>
| COMPUTED [BY] (<выражение>)
| GENERATED ALWAYS AS (<выражение>)
```

```
}
[DEFAULT {<литерал> | NULL}]
[NOT NULL]
[<ограничение столбца>]
[COLLATE <порядок сортировки>]

<тип данных> ::=
{ SMALLINT [<размерность массива>]
| INTEGER [<размерность массива>]
| BIGINT [<размерность массива>]
| FLOAT [<размерность массива>]
| DOUBLE PRECISION [<размерность массива>]
| {DECIMAL | NUMERIC} [(<точность> [, <масштаб>])]
[<размерность массива>]
| {DATE | TIME | TIMESTAMP} [<размерность массива>]
| {CHAR | CHARACTER | CHARACTER VARYING | VARCHAR}
[(<целое>)] [CHARACTER SET <набор символов>]
[<размерность массива>]
| {NCHAR | NATIONAL CHARACTER | NATIONAL CHAR}
[VARYING] [(<целое>)]}
[<размерность массива>]
| BLOB [SUB_TYPE {<номер подтипа> | <имя подтипа>}]
[SEGMENT SIZE <целое>]
[CHARACTER SET <набор символов>]}
| BLOB [(<размер сегмента> [, <номер подтипа>])]
}
<размерность массива> ::= [<целое 1>:<целое 2>
[, <целое 1>:<целое 2>]...]


<ограничение столбца> ::= [CONSTRAINT <имя ограничения>]
{ UNIQUE [<предложение USING>]
| PRIMARY KEY [<предложение USING>]
| REFERENCES <имя таблицы> [(<имя столбца>)]
[<предложение USING>]
[ON DELETE
{ NO ACTION
| CASCADE
| SET DEFAULT
| SET NULL
}
]
[ON UPDATE
{ NO ACTION
| CASCADE
| SET DEFAULT
| SET NULL
}
]
| CHECK (<условие столбца>)
}
<предложение USING> ::= USING
[ASC[ENDING] | DESC[ENDING]] INDEX <имя индекса>


<условие столбца> ::=
{ <значение> <оператор сравнения>
{<значение> | (<выбор одного>)}
| <значение> [NOT] IN
(<значение> [, <значение>]... | <поиск одного>})
| <значение> [NOT] BETWEEN <значение> AND <значение>
| <значение> [NOT] LIKE <значение> [ESCAPE '<символ>']
| <значение> IS [NOT] NULL
```

| <значение> IS [NOT] DISTINCT FROM <значение>
| <значение> <оператор сравнения>
{ALL | SOME | ANY} (<поиск одного>)
| EXISTS (<поиск многих>)
| SINGULAR (<поиск многих>)
| <значение> [NOT] CONTAINING <значение>
| <значение> [NOT] STARTING [WITH] <значение>
| (<условие столбца>)
| NOT <условие столбца>
| <условие столбца> OR <условие столбца>
| <условие столбца> AND <условие столбца>
}

Описание конструкций оператора CREATE TABLE представлено в таблице Б.2.

Таблица Б.1. – Описание конструкций оператора CREATE TABLE

Аргумент	Описание
<имя таблицы>	Имя таблицы должно быть уникальным среди имен таблиц базы данных, хранимых процедур и представлений, описанных в этой базе данных
EXTERNAL [FILE] "<спецификация файла >"	Объявляет что данные для создаваемой таблицы, постоянно располагаются во внешней к базе данных таблице или файле.
<имя столбца>	Имя для столбца таблицы, должно быть уникальным именем в таблице.
<тип данных>	SQL тип данных столбца.
COMPUTED [BY] (<выражение>)	Указывает, что значение столбца вычисляется по другим столбцам. Выражение должно возвращать одиночное значение, и не иметь тип массива или возвращать массив.
<выражение>	Любое арифметическое выражение допустимое для типа данных столбца.
COLLATE <порядок сортировки>	Определяет порядок сортировки для столбца. Порядок сортировки на уровне столбца отменяет порядок сортировки определенный на уроне домена.
DEFAULT	Определяет значение по умолчанию столбца, если оно не определяется каким-либо другим образом Значения: -literal: Вставляется указанная строка, числовое значение или дата. -NULL: Вводится значение NULL -USER: Вводится имя текущего пользователя. Столбец должен иметь тип, совместимый с текстовым Установка значению по умолчанию на уровне столбца отменяет значение по умолчанию на уровне домена
CONSTRAINT <имя ограничения>	Помещает именованное ограничение на таблицу или столбец. Если имя ограничения опущено, Firebird создает системное имя для ограничения.

Ограничение целостности CHECK

Синтаксис:

CHECK (<*условие столбца*>);

В условии можно пользоваться операторами сравнения, операторами NOT, BETWEEN, LIKE, IN, IS [NOT] NULL и прочими. У столбца может быть только одно условие CHECK.

Условие (<*условие столбца* >) должно принимать значение истинно, для того чтобы разрешить операцию добавления или изменения данных. < *условие столбца* > может проверять несколько значений, введенных в другие столбцы. Это ограничение на уровне таблицы.

Синтаксис оператора INSERT

INSERT INTO <object> [(col [, col ...])]
{VALUES (<val> [, <val> ...]) | <select_expr>}
[RETURNING <column_list> [INTO <variable_list>]];

<object> = tablename | viewname

val = {:variable | constant | expr
| function | udf ([val [, val …]])
| NULL | USER | RDB$DB_KEY | ?
} [COLLATE collation]

<constant> = num | 'string' | charsetname 'string'

<expr> = Допустимое выражение SQL, которое возвращает в одиночное значение столбца.

<function> = {
CAST (<val> AS <datatype>)
 | UPPER (<val>)
 | GEN_ID (generator, <val>)
 }

<select_expr> = SELECT возвращающий ноль или более строк,
 где число столбцов в каждой строке такое же,
 как число элементов, которые должны быть вставлены.

Описание конструкций оператора INSERT представлено в таблице Б.3.

Таблица Б.3 – Описание конструкций оператора INSERT

Аргумент	Описание
INTO <object>	Имя существующей таблицы или вида, в которую вставляются данные.
col	Имя существующего столбца в таблице или виде, в который вставляются значения.
VALUES (<val> [, <val> …]	Список значению для вставки в таблицу или вид. Значения должны быть в том же порядке, как целевые столбцы.
RETURNING <column_list> [INTO <variable_list>]	Только для Firebird 2.0 и выше.
<select_expr>	Запрос, который возвращает значения, для вставки в целевые столбцы.
<column_list>	Список возвращаемых доменов объекта, только Firebird v2.0 и выше
<variable_list>	Список переменных в которые будут помещены возвращаемые домены объекта. Только Firebird v2.0 и выше

Синтаксис оператора SELECT

<select statement> ::=
 <select expression> [FOR UPDATE] [WITH LOCK]

<select_expression> ::=
 <select_expr_body>
 [ORDER BY <sort_value_list>]
 [ROWS <record_number> [TO <record_number>]]

<select_expr_body> ::=
 { <query_specification> | <select_expr_body> UNION [DISTINCT | ALL] <query_specification> }

<query_specification> ::=
 SELECT
  [DISTINCT | ALL]
  [FIRST {<record_number> | (:param_recordnumber) } [SKIP <record_number> | (:param_recordnumber)]]
  <select_list>
 FROM <reference_expression_list>
 [WHERE <search condition>]
 [GROUP BY <group_value_list>]
 [HAVING <group_condition>]
 [PLAN <plan_item_list>]

<select_list> ::=
 { <constant>
 | <column_name>[<array_dim>]
 | <expression>
 | <function> ([<select_list>])
 | <UDF> ([<select_list>])
 | NULL

```
  | {CURRENT_USER | CURRENT_ROLE | CURRENT_TIMESTAMP | CURRENT_CONNECTION |
CURRENT_XXX }
  | RDB$DB_KEY
  } [COLLATE <collate>] [AS <column_alias_name>]

<reference_expression_list> ::=
  {<table name> | <view name> | <procedure name> | <joined table> | <derived table> }

<derived table> ::=
  '(' <select expression> ')'

<joined table> ::=
  { <cross_join> | <qualified_join> }

<cross_join> ::=
  <reference_expression_list> CROSS JOIN <reference_expression_list>

<qualified_join> ::=
  <reference_expression_list> [{INNER | {LEFT | RIGHT | FULL} [OUTER]}] JOIN <reference_expression_list>
ON <join condition>

<search_condition> ::=
    <val> <operator> { <val> | <singletone_select> }
  | <val> [NOT] BETWEEN <val> AND <val>
  | <val> [NOT] LIKE <val> [ESCAPE <val>]
  | <val> [NOT] CONTAINING <val>
  | <val> [NOT] STARTING WITH <val>
  | <val> [NOT] IN ( { <val> [,<val> ] | <select expression> } )
  | <val> IS [NOT] NULL
  | {ALL | SOME | ANY} (<select expression>)
  | EXISTS (<select expression>)
  | SINGULAR (<select expression>)
  | {AND | OR} [NOT] (<search condition>)

<operator> ::=
  { = | < | > | <= | >= | !< | !> | <> | != }

<sort_value_list> ::=
  {<column_name> | <column_number | expression> } [COLLATE <collation> ] [ASC[ENDING] |
DESC[ENDING] [NULLS {FIRST|LAST}] ] [, <sort_value_list>]
```

Описание конструкций оператора SELECT представлено в таблице Б.4.

Таблица Б.4 – Описание конструкций оператора SELECT

Аргумент	Описание
SELECT [DISTINCT | ALL]	Определяет данные, для поиска. DISTINCT удаляет повторяющиеся значения из возвращенных данных. ALL, параметр по умолчанию, возвращает все данные.
*	возвращает все столбцы из определенной таблицы.
<select_list>	возвращает определенный список столбцов и значений
FROM <reference_expression_list>	Список таблиц, представлений и сохраненных процедур, из которых возвращаются данные. Список может включать JOIN, соединения могут быть вложенными.
<table name>	Имя существующей таблицы в базе данных.
<view name>	имя существующего представления в базе данных.
<procedure name>	Имя существующей сохраненной процедуры, которая функционирует, как инструкция SELECT.
<joined table>	Ссылка таблицы состоящая из JOIN.
WHERE <search_cond>	Определяет условия, которые ограничивают подмножество возвращаемых строк из всех доступных строк.
GROUP BY <group_value_list>	Разделяет результаты запроса в группы содержащие все строки с одинаковыми значениями, основанными на списке столбцов.
HAVING <group_condition >	Используется совместно с GROUP BY. Определяет условия, которые ограничивают группировку возвращаемых строк.

UNION	Комбинирует две или более таблиц, которые имеют полностью, либо частично одинаковую структуру.
ORDER BY <sort_value_list>	Определяет порядок в котором строки будут возвращены.

Оператор ALTER TABLE

Используется для модификации структуры существующей таблицы

Перед тем как использовать ALTER TABLE:

- сохраните данные таблицы
- удалите все ограничения целостности со столбца

Сохранение данных это процесс, состоящий из пяти шагов. Например, один из столбцов таблицы имеет тип CHAR(3), его надо поменять на CHAR(5).

Для этого надо:

1. Создать временный столбец, определив его так же, как существующий.
2. Переписать данные из существующего столбца во временный столбец
3. Модифицировать временный столбец
4. Переписать данные из временного столбца обратно
5. Уничтожить временный столбец

1. Добавляем новый столбец
   ALTER TABLE EMPLOYEE ADD TEMP_NUM CHAR(3);
2. Переписываем
   UPDATE EMPLOYEE SET TEMP_NUM = DEP_NUM;
3. Модифицируем
   ALTER TABLE ALTER DEP_NUM TYPE CHAR(4);
4. Переписываем обратно
   UPDATE EMPLOYEE SET DEP_NUM = TEMP_NUM;
5. Удаляем
   ALTER TABLE DROP TEMP_NUM;

Удаление столбца

ALTER TABLE <имя таблицы> DROP <имя столбца>[, <имя столбца>...];
Например:
ALTER TABLE EMPLOYEE
DROP EMP_NUM,
DROP JOB_CODE;

Команда ALTER TABLE не сможет выполниться при следующих условиях:

- если данные в таблице после удаления нарушают ограничения PRIMARY KEY или UNIQUE
- если столбец является частью первичного ключа, ограничения UNIQUE, или столбец является внешним ключом
- столбец используется в ограничении CHECK
- столбец используется в представлении (view), в триггере, или используется при вычислении другого поля.

В случае, если что-либо мешает удалить столбец, надо сначала удалить это «что-то», а затем удалять столбец.

Добавление столбца

ALTER TABLE <имя таблицы> ADD <определение столбца>
Например:
ALTER TABLE employee ADD emp_no INTEGER NOT NULL;

Добавление ограничения целостности на таблицу

ALTER TABLE <имя таблицы> ADD [CONSTRAINT <имя ограничения>] <определение ограничения>;
Можно добавить ограничения PRIMARY KEY, FOREIGN KEY, UNIQUE, или CHECK.
Например:
ALTER TABLE EMPLOYEE ADD CONSTRAINT DEPT_NO UNIQUE(PHONE_EXT);

Удаление ограничения целостности, наложенного на столбец

ALTER TABLE <имя таблицы>
DROP CONSTRAINT <имя ограничения>;

Например:
ALTER TABLE PROJECT
DROP CONSTRAINT TEAM_CONSTRT;

Если имя ограничения небыло указано явно при создании таблицы, сервер генерирует имя автоматически. Посмотреть имена ограничений можно в системной таблице RDB$RELATION_CONSTRAINTS:
SELECT * FROM RDB$RELATION_CONSTRAINTS

Модификация столбца в таблице:

Можно изменить имя столбца, тип столбца, позицию столбца в таблице.
ALTER TABLE <имя таблицы> ALTER [COLUMN] <имя столбца> <определение изменеия>

<определение изменеия>:
- для имзменеия имени: TO <новое имя>
- для изменеия типа данных: TYPE <новый тип>
- для изменения позиции: POSITION <целое число>

Примеры:
1. Изменение позиции
   ALTER TABLE EMPLOYEE ALTER EMP_NUM POSITION 2;
2. Изменение имени EMP_NUM на EMP_NO
   ALTER TABLE EMPLOYEE ALTER EMP_NUM TO EMP_NO;
3. Изменение типа
   ALTER TABLE EMPLOYEE ALTER EMP_NO TYPE CHAR(20);

Преобразование из не-символьных в символьные типы возможны при следующих ограничениях.
- BLOB и массивы не преобразуются.
- Новое определение поля должно вмещать старые данные (CHAR(3) не вместит CHAR(4))