aba80b84033c9ef196388e9ea0434f90a83a40954ba270e84bcbe64f02d1d8d0

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное бюджетное образовательное учреждение высшего образования «Казанский национальный исследовательский технический университет им. А.Н. Туполева-КАИ»
(КНИТУ-КАИ)
Чистопольский филиал «Восток»

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ПРАКТИЧЕСКИМ РАБОТАМ

по дисциплине

БАЗЫ ДАННЫХ

Индекс по учебному плану: **Б1.О.18**
Направление подготовки: **09.03.01 Информатика и вычислительная техника**
Квалификация: **Бакалавр**
Профиль подготовки: **Автоматизированные системы обработки информации и управления**
Типы задач профессиональной деятельности: **проектный, производственно-технологический**

Рекомендовано УМК ЧФ КНИТУ-КАИ

Чистополь
2023 г.

ОПЕРАЦИИ НАД ОТНОШЕНИЯМИ

Одним из основных преимуществ реляционной модели является ее однородность. Все данные рассматриваются как хранимые в таблицах, в которых каждая строка имеет один и тот же формат.

Схемой отношения R называется конечное множество имен атрибутов $\{A_1,A_2,...,A_N\}$. Каждому множеству атрибута A_i ставится в соответствие множество D_i, называемое доменом атрибута A_i, $1 \leq i \leq N$. Домены являются произвольными непустыми конечными или счетными множествами.

Обновление отношений.Что можно делать с отношениями? Содержимое отношений изменяется во времени. Предположим, что нужно поместить дополнительную информацию в отношение. Для этого вводится операция добавление, которая для отношения $R(A_1,A_2,... ,A_N)$ имеет вид:

ADD (R; $A_1=d_1$, $A_2=d_2$,.. .,$A_N=d_N$)

Пример.

ADD (Расписание; НОМЕР=117, ПУНКТ_ОТПРАВЛЕНИЯ=РИГА,
ПУНКТ_НАЗНАЧЕНИЯ=КИЕВ, ВРЕМЯ_ВЫЛЕТА=22.05,
ВРЕМЯ_ПРИБЫТИЯ=0.43)

Когда порядок имен атрибутов фиксирован, допустима более короткая запись:

ADD (Расписание; 117, РИГА, КИЕВ, 22.05, 0.43)

Результат операции может быть ошибочен, если: добавляемый кортеж не соответствует схеме определенного отношения; некоторые значения кортежа не принадлежат соответствующим доменам; описанный кортеж совпадает по ключу с кортежем, уже находящимся в отношении. Во всех случаях операция ADD оставляет отношение R неизменным и сообщает об ошибке.

Операция удаление:

DEL (R; $A_1=d_1$, $A_2=d_2$, ..., $A_N=d_N$)

Если имена атрибутов упорядочены, то DEL (R; d_1, d_2, ..., d_i). Например:

DEL (расписание, 305, МОСКВА, СВЕРДЛОВСК, 21.50, 15.10)

Можно произвести удаление по ключу: DEL (расписание; 83).

Операция изменения:

CH (R; $B_1=d_1$; $B_2=d_2$,..., $B_m=d_m$; $C_1=e_1$, $C_2=e_2$, ..., $C_p=e_p$)

Пример.

CH (Расписание; НОМЕР=323, ПУНКТ_ОТПРАВЛЕНИЯ=МОСКВА,
ПУНКТ_НАЗНАЧЕНИЯ=СВЕРДЛОВСК, ВРЕМЯ_ВЫЛЕТА=21.30, ВРЕМЯ_ПРИБЫТИЯ= 14.50)

Сокращенный вариант:

CH (расписание; ОМЕР=323, ВРЕМЯ_ВЫЛЕТА=21.30, ВРЕМЯ_ПРИБЫТИЯ=14.50)

5.1. Выполнение операций над отношениями

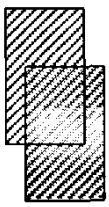
Для получения информации из отношений необходим язык манипулирования данными (ЯМД), способный выполнять соответствующие операции над отношениями. Наиболее важной частью ЯМД является его раздел для формулировки запросов.

Были разработаны три типа теоретических языков: реляционная алгебра, реляционное исчисление с переменными-кортежами, реляционное исчисление с переменными-доменами. Реальные языки (ISBL, SEQUEL, QBE и другие).

Реляционная алгебра. Основные операции реляционной алгебры можно разразделить на традиционные и специализированные. Операции первой группы: декартово произведение, объединение, пересечение, разность, симметричная разность. Операции второй группы: проекция, ограничение, соединение, деление.

1. Объединение отношений R_1 и R_2

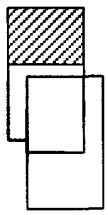
$$R = R_1 \cup R_2 = \{r \mid r \in R_1 \vee r \in R_2\}$$



Операция применяется только к отношениям одной и той же арности. Отношение R также той же арности.

2. Разность отношений R_1 и R_2

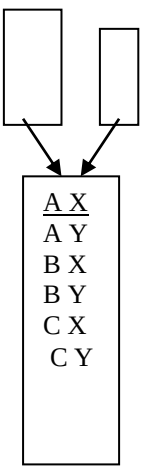
$$R = R_1 - R_2 = \{r \mid r \in R_1 \wedge r \notin R_2\}$$



Разностью $(R_1 - R_2)$ множество кортежей, принадлежащих R1, но не принадлежащих R_2. Отношения R_1 и R_2 должны быть одинаковой арности.

3. Декартово произведение отношений R_1 и R_2

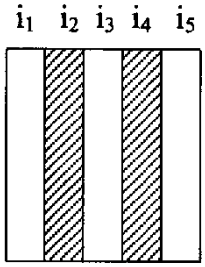
$$R = R_1 \times R_2 = \{r_1 r_2 \mid r_1 \in R_1 \wedge r_2 \in R_2\}$$



Если отношение R_1 имеет арность k_1, а отношение R_2 - арность k_2 то декартовым произведением отношений R_1 и R_2 является множество кортежей арности $(k_1 + k_2)$. Причем первые k_1 элементов образуют кортеж из отно - шения R_1, а последние k_2 элементов – из отношения R_2.

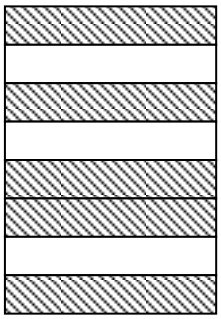
4. Проекция отношения R_1 на компоненты $i_1, i_2, ..., i_R$:

$R = \pi\, i_1, i_2, \ldots, i_R (R_1)$, где $i_1, i_2, ..., i_R$ – номера столбцов отношения R_1.



Операция проекция заключается в том, что из отношения R_1 выбираются указанные столбцы и компонуются в указанном порядке.

5.Селекция (выборка) отношения R_1 по формуле F:



$R = \sigma_F (R_1)$ или $R [A\ \theta\ B]$, где A и B – домены.

F – формула, образованная :

а) операндами, являющимися номерами столбцов;

б) логическими операторами $\wedge$ – и, $\vee$ – или, $\neg$ – не;

в) арифметическими операторами сравнения: $<$, $=$, $>$, $\leq$, $\neq$, $\geq$.

В формуле могут использоваться скобки.

На рис.5.1 приведены примеры операций реляционной алгебры над

R_1

а	б	в
д	е	м
и	к	л

R_2

и	к	л
г	д	е

а

$R=R_1 \vee R_2$

а	б	в
д	е	м
и	к	л
г	д	е

б

$R=R_1 - R_2$

а	б	в
д	е	м

в

$R=R_1 \times R_2$

а	б	в	и	к	л
д	е	м	и	к	л
и	к	л	и	к	л
а	б	в	г	д	е
д	е	м	г	д	е
и	к	л	г	д	е

г

$R=\pi_{1,2} (R_1)$

а	б
д	е
и	к

$R=\pi_{3,1} (R_1)$

в	а
м	д
л	и

д

$R=\sigma_{1=a \vee 2=e} (R_1)$

а	б	в
д	е	м

$R=\sigma_{2 = k} (R_2)$

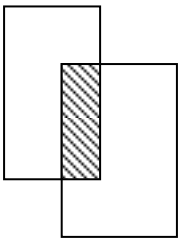
и	к	л

е

Рис.5.1.

а. Исходные отношения R_1 и R_2, б. Объединение отношений, в. Разность отношений , г. Декартово произведение, д. Проекция, е. Селекция

6. Пересечение отношений R_1 и R_2 (рис.5.2).



$$R = R_1 \cap R_2 = R_1 - (R_1 - R_2)$$

R_1

и	к	а
с	а	м
и	к	б
о	л	р
д	б	с

R_2

и	к	н
с	а	м
р	к	б
о	л	р
е	д	в

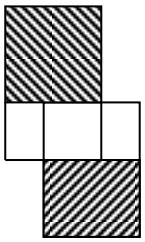
$R_1 \cap R_2$

с	а	м
о	л	р

Рис.5.2. Пересечение отношений R_1 и R_2

7. Частное отношение (симметричная разность) (рис.5.3)

$R = R_1 \div R_2 = \{ r \mid r \in R_1 \vee r \in R_2$,но не обоим вместе $\}$



R_1

а	б	с
д	е	к
л	м	н

R_2

д	е	к
р	с	т

$R = R_1 \div R_2$

а	б	с
л	м	н
р	с	т

Рис.5.3. Частное отношение

8. Соединение отношений (рис.5.4 и 5.5).

$$R = R_1 [A\ \theta\ B]\ R_2 = \{(r_1\ r_2) \mid (r_1 \in R_1) \wedge (r_2 \in R_2) \wedge r_1 [A]\ \theta\ r_2 [B]\}$$

$R = R_1 [A\ \theta\ B]\ R_2$, где A и B – атрибуты (имена атрибутов , а θ: $=, \neq, <, \leq, \geq, >$.

R_1

A	B	C
а	с	с
а	р	р
д	е	ж

R_2

Д	Е
а	и
е	к

$R = R_1 [B=Д]\ R_2$

A	B	C	Д	Е
д	е	ж	е	к

Рис.5.4. Пример соединения отношений

А1	б1
А2	б1
А3	б2

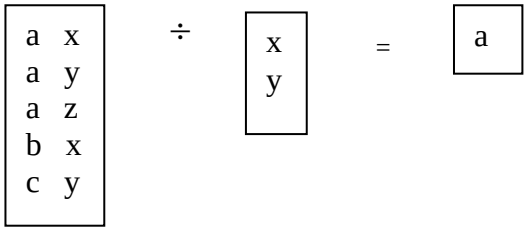
б1	с1
б2	с1
б3	с2

а1	б1	с1
а2	б1	с1
а3	б2	с1

Рис.5.5. Пример эквисоединения

9. Деление - представляет процесс, соответствующий операции обратной к декартовому произведению.

$$(R_1 \otimes R_2) \div R_2 = R_1$$



Достоинства реляционной модели: простота, наличие непроцедурных запросов, независимость данных.

Недостатки: более низкая производительность.

Практические задания

1.Дано отношение: Поставки (Код_поставщика, Код_товара, Наименование_поставщика. Адрес, Наименование_товара, Цена_товара). Произвести нормализацию данного отношения.

2. Спроектировать базу данных для получения сведений о студентах. Возможные атрибуты: Ном.факультета, Название, ФИО_декана, Телефон, Номер_группы, Специальность, Количество_студентов_в_группе,ФИО_старосты, Стипендия, ФИО_студента, Ном.зач.книжки.

3. Спроектировать базу данных о сотрудниках. Возможные атрибуты: ФИО, Год_рождения, Должность, Отдел, Домашний_адрес, ВУЗ, Адрес_вуза, Год_окончания, Специальность, Сведения_о_детях..

4. Дано отношение: ЭКЗАМЕНАЦИОННАЯ_ВЕДОМОСТЬ(Факультет, Группа, Дисциплина, Семестр, Учебный_год, ФИО_преподавателя, Вид_зачета, ФИО_студента, Номер_зачетной_книжки, Оценка). Произвести нормализацию и спроектировать базу данных.

5. Спроектировать базу данных о поставщиках и деталях. Возможные атрибуты: Наименование_поставщика, Адрес, Номер_счета_в_госбанке, Наименование_детали, Номер_Госта, Единица_измерения, Цена.

6. Дано отношение: ПОСТАВКИ (Номер_клиента, ФИО, Адрес, Номер_партии_товара, Название_товара, Цена, Учетный_номер, Количество). Произвести нормализацию и спроектировать базу данных.

7. По результатам сессии выявить задолжников по заданной группе. Возможные атрибуты: Номер_группы, Специальность, Староста, Куратор, ФИО_студента, Номер_зачетной_книжки, Дисциплина, Оценка. Спроектировать логическую базу данных.

8. Дано отношение: Наличие_лекарств_в_аптеках (Номер_аптеки, Адрес, Телефон, Номер_лекарства, Наименование, Стоимость, Вес_упаковки, Количество_лекарства_в_данной_аптеке). Произвести нормализацию и спроектировать базу данных.

9. Дано отношение: Поставки (Код_поставщика, Имя_поставщика, Адрес, Код_товара, Наименование_товара, Цена_товара, Единица_измерения_товара, Количество_поставленного_товара). Произвести нормализацию данного отношения.
10. Спроектировать базу данных о научных сотрудниках. Возможные атрибуты: ФИО, Год_рождения, Должность, Ученая_степень, Ученое_звание, Количество_опубликованных_работ, Перечень_работ, Сведения_о_соавторах_каждой работы).

РЕЛЯЦИОННЫЕ ЯЗЫКИ ЗАПРОСОВ

Существуют языки: ISBL - язык системы PRTV, QUEL -язык системы INGRES, SQL - язык системы System R; QBE - язык интерфейса высокого уровня в ряде СУБД, PIQUE - язык экспериментальной системы PITS. ISBL основан на реляционной алгебре. QUEL и SQL - на исчислении кортежей. QBE - язык, основанный на исчислении доменов. PIQUE - похож на исчисление кортежей, но обеспечивает интерфейс со схемой помощью W-функций.

6.1. Язык SQL (Structured Query Language)

Язык был разработан в компании IBM в начале 70-х годов, его первой реализацией был продукт System R; впоследствии он был реализован в многочисленных коммерческих продуктах как фирмы IBM, так и других изготовителей. Язык SQL стал Американским национальным стандартом (ANSI), Международным стандартом (ISO), стандартом системы UNIX (X/Open). Он используется для описания реляционных операций. Стандарты: SQL/89, SQL/92.

В реляционных системах выполняются как минимум 2 условия:

1. Данные воспринимаются пользователем как таблицы.
2. В распоряжении пользователя имеются операторы (например, для выборки данных), среди которых есть операторы SELECT, PROJECT и JOIN.

- Операция SELECT предназначена для извлечения определенных строк из таблицы.
- Операция PROJECT предназначена для извлечения определенных столбцов из таблицы.

- Операция JOIN предназначена для соединения двух таблиц на основе общих значений в общих столбцах.

Все эти операции могут быть сформулированы на языке SQL.

Пример 6.1. Пусть имеются 2 таблицы:

ОТДЕЛЫ (№ отдела, Наименование, Бюджет)

СЛУЖАЩИЕ (№ служащего, № отдела, Оклад)

1. SELECT:

```
SELECT № отдела, Наименование, Бюджет
FROM  ОТДЕЛЫ
WHERE Оклад >400р.
```

2 PROJECT:

```
SELECT № отдела, Бюджет
FROM   ОТДЕЛЫ
```

3. JOIN:

```
SELECT ОТДЕЛЫ.*, СЛУЖАЩИЕ.*
FROM  ОТДЕЛЫ, СЛУЖАЩИЕ
WHERE ОТДЕЛЫ. № отдела = СЛУЖАЩИЕ. № отдела
```

Функции определения данных:

CREATE TABLE - оператор создания таблицы ;

CREATE INDEX - оператор создания индексной таблицы;

DROP TABLE - оператор удаления таблицы ;

DROP INDEX - оператор удаления индексной таблицы.

6.2. Операторы манипулирования данными

В языке SQL предусмотрено четыре предложения манипулирования данными: SELECT (выдать, выбрать), UPDATE (обновить), DELETE (удалить) и INSERT (включить).

Создадим БД Поставщиков (S), Деталей (P) и Поставок (SP).

```
CREATE  TABLE  S
(НОМЕР_ПОСТАВЩИКА  CHAR (5),
ИМЯ          CHAR (20),
СОСТОЯНИЕ          SMALLINT,
ГОРОД              CHAR (15));
   CREATE TABLE P
(НОМЕР_ДЕТАЛИ      CHAR (6),
НАЗВАНИЕ_ДЕТАЛИ          CHAR (20),
ЦВЕТ             CHAR (7),
ВЕС              SMALLINT,
Город            CHAR (15));
    CREATE TABLE SP
       (НОМЕР_ПОСТАВЩИКА CHAR (5),
       НОМЕР_ДЕТАЛИ      CHAR (6),
       КОЛИЧЕСТВО        INTEGER);
```

Содержимое базы данных представлено в табл. 6.1, 6.2, 6.3

Поставщики (S) Таблица 6.1

НОМЕР_ПОСТАВЩИКА	ИМЯ	СОСТОЯНИЕ	ГОРОД
S1	Саша	20	Липецк
S2	Дима	10	Пермь
S3	Боря	30	Пермь
S4	Коля	20	Липецк
S5	Алик	30	Арзамас

Детали (P) Таблица 6.2

НОМЕР_ДЕТАЛИ	НАЗВАНИЕ	ЦВЕТ	ВЕС	ГОРОД
P1	Гайка	Красный	12	Липецк
P2	Болт	Зеленый	17	Пермь
P3	Винт	Голубой	17	Рига
P4	Винт	Красный	14	Липецк
P5	Кулачок	Голубой	12	Пермь
P6	Блюм	Красный	19	Липецк

Поставки (SP) Таблица 6.3

НОМЕР_ПОСТАВЩИКА	НОМЕР_ДЕТАЛИ	КОЛИЧЕСТВО
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

6.3.Выборки

Основной операцией в языке SQL является отображение, синтаксически представляющее собой блок SELECT – FROM - WHERE (выбрать – из – где).

Структура оператора SELECT:

```
SELECT [DISTINCT] элементы
FROM  таблица
[WHERE   предикат]
[GROUP BY поле [HAVING [предикат]]
[ORDER BY поле]
```

Пример 6.2. Выдать номера и состояния для поставщиков, находящихся в Перми:

```
SELECT НОМЕР_ПОСТАВЩИКА, СОСТОЯНИЕ
FROM  S
WHERE ГОРОД= 'Пермь'
```

Результат:

НОМЕР ПОСТАВЩИКА	СОСТОЯНИЕ
S2	10
S3	30

Пример 6.3. Простая выборка. Выдать номера для всех поставляемых деталей:

```
SELECT НОМЕР_ДЕТАЛИ
FROM  SP
```

Результат:

НОМЕР ДЕТАЛИ
P1
P2
P3
P4
P5
P6
P1
P2
P2
P4
P5

Пример 6.4. Выборка с исключением дубликатов.

```
SELECT DISTINCT НОМЕР_ДЕТАЛИ
FROM   SP
```

Результат:

НОМЕР ДЕТАЛИ
P1
P2
P3
P4
P5
P6

SELECT DISTINCT - выбрать различные.

Пример 6.5. Выборка вычисляемых значений. Выдать номер и вес каждой детали:

```
SELECT  НОМЕР_ДЕТАЛИ, ВЕС
FROM  P
```

Результат:

НОМЕР ДЕТАЛИ	ВЕС
P1	12
P2	17
P3	17
P4	14
P5	12
Pб	19

Пример 6.6. Фраза SELECT (и фраза WHERE) может включать арифметические выражения, а также прочие имена полей. Можно, кроме того, осуществлять выборку просто констант.

```
SELECT НОМЕР_ДЕТАЛИ, 'Вес в граммах=', ВЕС
FROM   P
```

Результат:

НОМЕР ДЕТАЛИ	ВЕС	
. P1	Вес в граммах=	12
P2	Вес в граммах=	17
P3	Вес в граммах=	17
P4 .	Вес в граммах=	14
P5	Вес в граммах=	12
P6	Вес в граммах=	19

Пример 6.7. Выдать полные характеристики для всех поставщиков:

```
SELECT *
FROM   S
```

Результатом будет таблица S со всеми полями.

Пример 6.8. Ограниченная выборка. Выдать номера поставщиков, которые находятся в Перми и имеют состояние больше, чем 20.

```
SELECT НОМЕР_ПОСТАВЩИКА
 FROM   S
WHERE ГОРОД='Пермь'  AND   СОСТОЯНИЕ > 20
```

Результат:

НОМЕР ПОСТАВЩИКА
S3

Условие или предикат, следующий за ключевым словом WHERE, может включать операторы сравнения =, ¬= (не равно), >, ¬>, >=, , ¬< и <=, булевские операторы AND (и), OR (или) и NOT (нет), а скобки указывают требуемый порядок вычислений.

Пример 6.9. Выборка с упорядочением. Выдать номера и состояния поставщиков, находящихся в Перми, в порядке убывания их состояния:

```
SELECT НОМЕР_ПОСТАВЩИКА
FROM  S
WHERE ГОРОД='Пермь'
ORDER BY СОСТОЯНИЕ DESC
```

Результат:

НОМЕР ПОСТАВЩИКА	СОСТОЯНИЕ
S3	30
S2	10

Параметры:

ASC – возрастание, DECS – убывание, ORDER BY – упорядочить.

При упорядочивании необходимо указать номер столбца:

```
SELECT НОМЕР_ДЕТАЛИ, ВЕС
FROM   P
ORDER BY 2, НОМЕР_ДЕТАЛИ
```

В строке ORDER BY … цифра 2 означает ссылку на второй столбец результирующей таблицы.

Результат:

НОМЕР_ДЕТАЛИ	ВЕС
P1	12
P5	12
P4	14
P2	17
P3	17
P6	19

Пример 6.10. Выборка с использованием BETWEEN (между). Выдать сведения о деталях, вес которых находится в диапазоне от 16 до 19 включительно:

```
SELECT НОМЕР_ДЕТАЛИ, НАЗВАНИЕ, ЦВЕТ, ВЕС, ГОРОД
FROM   P
WHERE ВЕС BETWEEN 16 AND 19
```

Результат:

НОМЕР ДЕТАЛИ	НАЗВАНИЕ	ЦВЕТ	ВЕС	ГОРОД
P2	Болт	Зеленый	17	Пермь
P3	Винт	Голубой	17	Рига
P6	Шайба	Красный	19	Липецк

Может также использоваться NOT BETWEEN.

Пример 6.11. Выборка с использованием IN .Выдать детали, вес которых 12,16,17.

```
SELECT  НОМЕР _ДЕТАЛИ, НАЗВАНИЕ, ВЕС, ГОРОД
FROM    P
WHERE  ВЕС IN (12, 16,17)
```

Существует также предикат NOT IN (не принадлежит).

Пример 6.12. Выборка с использованием предиката LIKE (похоже на).Выдать все детали, название которых начинается с буквы В:

```
SELECT НОМЕР_ДЕТАЛИ, НАЗВАНИЕ, ЦВЕТ, ВЕС, ГОРОД
 FROM   P
 WHERE  НАЗВАНИЕ LIKE 'В%'
```

Обычно предикат LIKE имеет форму:

Имя столбца LIKE литерная строковая константа,

где имя столбца имеет тип CHAR.

Литеры:

- Литера " _ " (разрыв или подчеркивание) - обозначает любую одиночную литеру.
- Литера " %" (процент) - обозначает любую последовательность из п литер.
- Все другие литеры обозначают просто сами себя.

Примеры:

АДРЕС LIKE '% Брест %' – будет приниматься истина, если АДРЕС содержит строку "Брест".

НОМЕР_ПОСТАВЩИКА LIKE 'S_ _' – будет истина, если номер состоит из трех литер и первая из них литера S.

НАЗВАНИЕ LIKE '% К_ _ _' – будет истина, если название состоит из четырех и более литер и трем последним из них предшествует литера К.

ГОРОД NOT LIKE '% Е %' – будет истина, если значение литера не содержит литеры Е.

6.4.3апросы, использующие соединения

Способность соединять две или более таблицы в одну, представляет собой одну из наиболее мощных средств реляционных систем.

Пример 6.13. Простое эквисоединение. Выдать все комбинации информации о таких поставщиках и деталях, которые размещены в одном и том же городе:

```
SELECT S.*,P.*
FROM   S,P
WHERE S.ГОРОД == P.ГОРОД
```

Будут выбраны все строки с одинаковыми городами. Это декартово произведе– ние таблиц.

Варианты:

- WHERE S.ГОРОД > P.ГОРОД
- WHERE S.ГОРОД = P.ГОРОД
  AND S.СОСТОЯНИЕ ⎤=20

При соединении необязательно чтобы было равенство. В случае равенства соединение называется эквисоединением.

В SELECT могут быть указаны выборочные поля, а не все. Выражение:

```
SELECT S.*, P.*          может быть записано:     SELECT  *
FROM   S,P                                        FROM   S,
```

Но может быть записано и с указанием всех полей.

Так как при соединении возникают два одинаковых столбца (S.Город и P.Город), то один столбец исключается и такое соединение называется естест- венным. Могут быть соединения и 3, 4 или любого числа таблиц.

Пример 6.14. Соединение с дополнительным условием. Выдать все комбинации информации о поставщиках и деталях, такие, что рассматриваемые поставщи – ки и детали "соразмещены". Опустить при этом поставщиков с состоянием 20:

```
SELECT *
FROM   S, P
WHERE S.ГОРОД = P.ГОРОД
AND      S.СОСТОЯНИЕ ¬=20
```

Результат:

НОМЕР_ПОСТАВЩИКА	ИМЯ	СОСТОЯНИЕ	S.ГОРОД
S2	Дима	10	Пермь
S2	Дима	10	Пермь
S3	Боря	30	Пермь
S3	Боря	30	Пермь

НОМЕР_ДЕТАЛИ	НАЗВАНИЕ	ЦВЕТ	ВЕС	Р.ГОРОД
P2	Болт	Зеленый	17	Пермь
P5	Кулачок	Голубой	12	Пермь
P2	Болт	Зеленый	17	Пермь
P5	Кулачок	Голубой	12	Пермь

6.5.Подзапросы

Подзапросы представляют собой вложенные предложения SELECT. Именно такая возможность позволила назвать язык SQL структурированным.

Пример6.15. Выдать фамилии поставщиков, поставляющих детали P2:

```
SELECT      ИМЯ
FROM        S
WHERE       НОМЕР_ПОСТАВЩИКА   IN
           (SELECT      НОМЕР_ПОСТАВЩИКА
            FROM        SP
            WHERE       НОМЕР_ДЕТАЛИ = 'P2');
```

Результат:

ИМЯ
Саша
Дима
Боря
Коля

Этот подзапрос возвращает множество поставщиков, поставляющих деталь P2: S1, S2, S3, S4. Поэтому первоначальный запрос эквивалентен простому запросу :

```
SELECT     ФАМИЛИЯ
FROM       S
WHERE      НОМЕР_ПОСТАВЩИКА  IN ('S1','S2','S3','S4')
```

Номер поставщика здесь задан неявно. Для явного задания необходимо указать имя таблицы.

```
SELECT S.ИМЯ
FROM   S
WHERE  S.НОМЕР_ПОСТАВЩИКА  IN
              (SELECT SP.НОМЕР_ПОСТАВЩИКА
               FROM    SP
              WHERE   SP.НОМЕР_ДЕТАЛИ='P2')
```

Этот же подзапрос может быть выражен и соединением:

```
SELECT S.ИМЯ
FROM    S, SP
WHERE  S.НОМЕР_ПОСТАВЩИКА = SP.НОМЕР_ПОСТАВЩИКА
AND     SP.НОМЕР_ДЕТАЛИ = 'P2'
```

6.6. Подзапросы с несколькими уровнями вложения

Пример 6.16. Выдать имена поставщиков, поставляющих красные детали:

```
SELECT  ИМЯ
FROM    S
WHERE  НОМЕР_ПОСТАВЩИКА  IN
            (SELECT НОМЕР_ПОСТАВЩИКА
             FROM  SP
             WHERE   НОМЕР_ДЕТАЛИ   IN
                        (SELECT  НОМЕР_ДЕТАЛИ
                         FROM    P
                          WHERE   ЦВЕТ = 'Красный'))
```

Результат:

ИМЯ
Саша
Дима
Коля

6.7. Коррелированный подзапрос.

Пример 6.17. Выдать имена поставщиков, которые поставляют деталь P2:

```
SELECT ФАМИЛИЯ
 FROM    S
 WHERE  ‘P2’  IN
          (SELECT НОМЕР_ДЕТАЛИ
           FROM    SP
           WHERE    НОМЕР_ПОСТАВЩИКА = S. НОМЕР_ПОСТАВЩИКА)
```

Система проверяет первую строку таблицы S. Предположим, что это строка поставщика "S1". Тогда переменная S.НОМЕР_ПОСТАВЩИКА в данный момент имеет значение "S1", и система обрабатывает внутренний запрос:

(SELECT НОМЕР_ДЕТАЛИ
FROM SP
WHERE НОМЕР_ПОСТАВЩИКА = ' S 1')

Далее система будет повторять обработку такого рода для следующего поставщика и т. д., пока не будут рассмотрены все строки таблицы S.

Такой подзапрос, как в этом примере, называется коррелированным. Для того чтобы сделать более ясной связь коррелированных подзапросов с внешними запросами, иногда вводят псевдонимы.

Например:

```
SELECT SX.ИМЯ
FROM S SX
WHERE 'P2' IN
      (SELECT НОМЕР_ДЕТАЛИ
      FROM SP
      WHERE НОМЕР_ПОСТАВЩИКА = SX. НОМЕР_ПОСТАВЩИКА)
```

Здесь псевдонимом является имя SX, введенное во фразе FROM как альтернативное имя таблицы S, т. е.

- SX - это переменная, областью определения которой является множество записей таблицы S.

- Поочередно для каждого возможного значения SX выполняется следующее:

1) вычисляется подзапрос и получается множество номеров деталей, P;

2) добавляется к результирующему множеству значение SX. ИМЯ, если только P2 принадлежит множеству P.

Некоторые считают более ясным использование двух различных символов для того, чтобы различать эти две различные функции.

Пример 6.18. Использование одной и той же таблицы в подзапросе и внешнем запросе. Выдать номера поставщиков, которые поставляют по крайней мере одну деталь, поставляемую поставщиком S2:

```
SELECT DISTINCT НОМЕР_ПОСТАВЩИКА
 FROM SP
 WHERE НОМЕР_ДЕТАЛИ IN
       (SELECT НОМЕР_ДЕТАЛИ
        FROM SP
        WHERE НОМЕР_ПОСТАВЩИКА = 'S2')
```

Результат: НОМЕР ПОСТАВЩИКА

S1
S2
S3
S4

Решение этой задачи с использованием псевдонимов:

```
SELECT DISTINCT SPX.НОМЕР_ПОСТАВЩИКА
FROM    SP    SPX
WHERE   SPX.НОМЕР_ДЕТАЛИ   IN
              (SELECT SPY.НОМЕР_ДЕТАЛИ
                FROM    SP      SPY
                WHERE   SPY.НОМЕР_ПОСТАВЩИКА = ' S2')
```

Эквивалентный запрос с использованием соединения имеет вид:

```
SELECT DISTINCT SPX.НОМЕР_ПОСТАВЩИКА
 FROM SP SPX, SP SPY
 WHERE SPX.НОМЕР_ДЕТАЛИ=SPY.НОМЕР_ДЕТАЛИ
 AND    SPY.НОМЕР_ПОСТАВЩИКА = ' S2'
```

Пример 6.19. Случай, когда в коррелированном и внешнем запросе используется одна и та же таблица. Выдать номера всех деталей, поставляемых более чем одним поставщиком:

```
 SELECT DISTINCT SPX.НОМЕР_ДЕТАЛИ
 FROM   SP  SPX
  WHERE SPX.НОМЕР_ДЕТАЛИ IN
             (SELECT SPY.НОМЕР_ДЕТАЛИ
               FROM    SP SPY
               WHERE SPY.НОМЕР_ПОСТАВЩИКА ¬=
                        SPX.НОМЕР_ПОСТАВЩИКА)
```

Результат:

НОМЕР ДЕТАЛИ
P1
P2
P4
P5

Пример 6.20. Подзапрос с оператором сравнения, отличным от IN. Выдать номера поставщиков, находящихся в том же городе, что и поставщик S 1:

```
SELECT НОМЕР_ПОСТАВЩИКА
 FROM   S
 WHERE ГОРОД=
             (SELECT ГОРОД
               FROM   S
               WHERE НОМЕР_ПОСТАВЩИКА = ' S 1')
```

Результат:

НОМЕР ПОСТАВЩИКА
Sl
S4

6.8. Квантор существования. Запрос, использующий EXISTS

Пример 6.21. Выдать фамилии поставщиков, которые поставляют деталь P2:

```
SELECT ИМЯ
FROM  S
WHERE  EXISTS
        (SELECT *
        FROM    SP
        WHERE    НОМЕР_ПОСТАВЩИКА  =  S.НОМЕР_ПОСТАВЩИКА
         AND  НОМЕР_ДЕТАЛИ='P2')
```

EXISTS (существует) представляет здесь квантор существования - понятие, заимствованное из формальной логики.

Пример 6.22. Запрос, использующий NOT EXISTS. Выдать имена поставщиков, которые не поставляют деталь P2:

```
SELECT ИМЯ
 FROM    S
WHERE  NOT EXISTS
          (SELECT *
           FROM    SP
            WHERE  НОМЕР_ПОСТАВЩИКА  =  S.НОМЕР_ПОСТАВЩИКА
           AND  НОМЕР_ДЕТАЛИ = 'P2')
```

6.9. Стандартные функции

В запросах могут использоваться следующие стандартные функции: COUNT – число значений в столбце; SUM – сумма значений какого-либо столбца; AVG – среднее значение; MAX – самое большое значение в столбце; MIN – самое малое значение в столбце.

Пример 6.23. Выдать общее количество поставщиков:

```
SELECT  COUNT (*)
 FROM    S
```

Результат: 5

Пример 6.24. Выдать общее количество поставляемых деталей P2:

```
SELECT SUM
 FROM    SP
WHERE   НОМЕР_ДЕТАЛИ = ‘P2’
```

Результат: 1000

Пример 6.25. Функция в подзапросе. Выдать номера поставщиков со значением поля СОСТОЯНИЕ меньшим, чем текущее максимальное состояние в таблице S:

```
SELECT  НОМЕР_ПОСТАВЩИКА
FROM   S
WHERE СОСТОЯНИЕ <
              (SELECT MAX (СОСТОЯНИЕ)
               FROM   S)
```

Результат:

НОМЕР ПОСТАВЩИКА
S1
S2
S3

6.10. Использование группировок (GROUP BY)

Пример 6.26. Вычислить общий объем поставок для каждой детали:

```
SELECT НОМЕР_ДЕТАЛИ, SUM (КОЛИЧЕСТВО)
 FROM   SP
GROUP   BY  НОМЕР_ДЕТАЛИ
```

Результат:

НОМЕР ДЕТАЛИ	
P1	600
P2	1000
P3	400
P4	500
P5	500
P6	100

Пример6.27. Запрос с использованием HAVING (Исключение всех групп, для которых не выполняется заданное условие). Выдать номера деталей, поставляемых более чем одним поставщиком:

```
SELECT НОМЕР_ДЕТАЛИ
FROM   SP
GROUP   BY НОМЕР_ДЕТАЛИ
HAVING COUNT (*) > 1
```

Результат:

НОМЕР ДЕТАЛИ
P1
P2
P4
P5

6.11. Объединение с использованием UNION

Пример 6.28. Выдать номера деталей, которые имеют вес больше 16, либо поставляются поставщиком S2 (либо то и другое):

```
SELECT НОМЕР_ДЕТАЛИ
FROM   P
WHERE ВЕС > 16
UNION
SELECT НОМЕР_ДЕТАЛИ
```

```
FROM   SP
WHERE НОМЕР_ПОСТАВЩИКА = 'S2'
```
Результат: НОМЕР ДЕТАЛИ
P1
P2
P3
P6

6.12. Многоаспектный запрос

Пример 6.29. Выдать номер детали, вес, цвет и максимальный объем поставки для всех красных и голубых деталей, таких, что общий объем их поставки больше, чем 350, исключая при этом из общего объема такие поставки, количество которых меньше или равно 200 деталей. Результат упорядочить по убыванию номеров деталей:

```
SELECT  P.НОМЕР_ДЕТАЛИ, 'Вес', P.ВЕС, P.ЦВЕТ, 'Максимальный объем
поставки =',    MAX (SP.КОЛИЧЕСТВО)
FROM   P, SP
WHERE P.НОМЕР_ДЕТАЛИ = SP.НОМЕР_ДЕТАЛИ
AND     P.ЦВЕТ IN ('Красный', 'Голубой')
AND     SP.КОЛИЧЕСТВО > 200
GROUP  BY  P.НОМЕР_ДЕТАЛИ, P.ВЕС, P.ЦВЕТ
HAVING SUM (КОЛИЧЕСТВО) > 350
ORDER BY P.НОМЕР_ДЕТАЛИ DESC
```
Результат:

PI	12	Красный	Максимальный объем поставки = 350
P5	12	Голубой	Максимальный объем поставки = 400
P3	17	Голубой	Максимальный объем поставки = 400

6.13. Операции обновления

UPDATE, DELETE, INSERT.
Предложение UPDATE:

```
UPDATE таблица
SET      поле = выражение [, поле = выражение]
[WHERE предикат]
```

Пример 6.30. Обновление одной записи. Изменить цвет детали P2 на желтый, увеличить ее вес на 5 и установить значение города "неизвестный" (NULL):

```
UPDATE P
SET     ЦВЕТ = 'Желтый',
        ВЕС = ВЕС + 5,
        ГОРОД = NULL
 WHERE НОМЕР_ДЕТАЛИ = 'P2'
```

Пример 6.31. Обновление множества записей. Удвоить состояние всех поставщиков, находящихся в Перми:

```
UPDATE S
SET    СОСТОЯНИЕ = 2 * СОСТОЯНИЕ
WHERE ГОРОД = 'Пермь'
```

Пример 6.32. Обновление с подзапросом. Установить нулевой объем поставок для всех поставщиков из Перми:

```
UPDATE SP
SET     КОЛИЧЕСТВО = 0
        WHERE       'Пермь' =
            (SELECT ГОРОД

           FROM   S
            WHERE S.НОМЕР_ПОСТАВЩИКА = SP. НОМЕР_ПОСТАВЩИКА)
```

Пример 6.33. Обновление нескольких таблиц. Изменить номер поставщика S2 на S9:

```
UPDATE S
SET    НОМЕР_ПОСТАВЩИКА = 'S9'
WHERE НОМЕР_ПОСТАВЩИКА = 'S2'
UPDATE SP
SET     НОМЕР_ПОСТАВЩИКА ='S9'
WHERE НОМЕР_ПОСТАВЩИКА = 'S2'
```

Здесь БД становится противоречивой после выполнения первой строчки UPDATE (нарушается целостность). Поэтому требуется второй UPDATE.

Предложение DELETE:

```
DELETE
FROM  таблица
[WHERE предикат]
```

Пример 6.34. Удаление единственной записи. Удалить поставщика S 1:

```
DELETE
FROM S
WHERE НОМЕР_ПОСТАВЩИКА = ' S 1'
```

Пример 6.35. Удаление множества записей. Удалить всех поставщиков из Перми:

```
  DELETE
  FROM S
   WHERE ГОРОД = 'Пермь'
```

Пример 6.36. Удалить все поставки:

```
DELETE
FROM SP
```

Пример 6.37. Удаление с подзапросом. Удалить все поставки для поставщиков из Риги:

```
DELETE
FROM SP
 WHERE ‘Рига’=
     (SELECT ГОРОД
       FROM  S
       WHERE S .НОМЕР_ПОСТАВЩИКА = SP.НОМЕР_ПОСТАВЩИКА)
```

Предложение INSERT:

```
INSERT
INTO     таблица [(поле [, поле] ...)]
VALUES  (константа [, константа] ...)
```

i – я константа соответствует i-му полю.

Пример 6.38. Вставка единственной записи.

```
INSERT
INTO    P (НОМЕР_ДЕТАЛИ, ГОРОД, ВЕС)
VALUES ('P5', 'Пермь',12)
```

Можно с опущенными именами полей:

```
INSERT
INTO    P
VALUES ('P5', 'Кулачок', 'Голубой', 12, 'Пермь')
```

Пример 6.39. Вставка множества записей. Для каждой поставляемой детали получить ее номер и общий объем поставок, сохранить результат в БД:

```
CREATE TABLE ВРЕМЕННАЯ
                (НОМЕР_ДЕТАЛИ CHAR(6),
                   ОБЪЕМ_ПОСТАВКИ INTEGER);
INSERT
INTO    ВРЕМЕННАЯ (НОМЕР_ДЕТАЛИ, ОБЪЕМ_ПОСТАВКИ)
SELECT НОМЕР_ДЕТАЛИ, SUM (КОЛИЧЕСТВО)
FROM   SP
GROUP BY  НОМЕР_ДЕТАЛИ
```

Здесь предложение SELECT выполняется точно так же, как обычно, но результат не возвращается пользователю, а копируется в таблицу ВРЕМЕННАЯ.

6.14. Представления

Представление - это виртуальная таблица, т. е. таблица, которая сама по себе не существует, но для пользователя выглядит, как будто она существует. В противоположность этому базовая таблица – это реальная таблица.
Синтаксис:

```
CREATE VIEW имя
        [(имя_столбца [, имя_столбца] ...)]
        AS подзапрос
```

Пример 6.40.

```
CREATE VIEW ХОРОШИЕ_ПОСТАВЩИКИ
 AS SELECT НОМЕР_ПОСТАВЩИКА, СОСТОЯНИЕ, ГОРОД FROM   S
WHERE СОСТОЯНИЕ > 15
```

Пример 6.41.

```
CREATE  VIEW  PQ (НОМЕР_ДЕТАЛИ, ОБЩЕЕ_КОЛИЧЕСТВО)
 AS SELECT НОМЕР_ПОСТАВЩИКА, SUM (КОЛИЧЕСТВО)
 FROM    SP
 GROUP   BY НОМЕР_ДЕТАЛИ
```

Здесь стандартная функция образует новое поле – ОБЩЕЕ_КОЛИЧЕСТВО. Далее может последовать запрос:

```
SELECT *
FROM   S
WHERE ОБЩЕЕ_КОЛИЧЕСТВО > 500
```

Для чего нужны представления? Одна из задач - обеспечение логической независимости данных. Система обеспечивает логическую независимость данных, если программы пользователей независимы от логической структуры БД. Имеются два аспекта такой независимости - рост и реструктуризация.

Рост - когда БД растет в связи с включением новых видов информации (расширение существующей базовой таблицы с включением нового поля, включение новой базовой таблицы). Реструктуризация - при изменении размещения некоторых полей в БД.

Преимущества представлений:

1. Обеспечивается определенная степень логической независимости.
2. Дается возможность различным пользователям по-разному видеть одни и те же данные. Упрощается пользовательское восприятие. Механизм представлений дает возможность пользователям сосредоточить внимание на данных, которые представляют для них интерес, и игнорировать остальные данные.
3. Обеспечение секретности для "скрытых данных " через механизм представления.

SQL - интерфейс.

Пользователю необходимо работать с различными СУБД, не изучая каждый раз язык запросов. Поэтому необходим единый язык, который бы входил в состав каждой СУБД. Этот язык должен быть непроцедурным. Таков язык SQL. SQL имеет недостатки:

- не позволяет писать функции и подпрограммы;
- не позволяет описать логику работы;
- имеет сложный, не очень понятный синтаксис;
- нет средств создания экранных форм и отчетов.

Практические задания

1. С помощью языка SQL произвести прибавку в размере 15тысяч рублей к стипендии студента Петрова

2. Дано отношение. Сотрудники (ФИО, Год_рожд., Должность, Отдел, Вуз, Адрес_вуза, Год_окончания, Специальность)
   На языке SQL получить сведения о сотрудниках, имеющих высшее образование и окончивших вуз до 1992 года.
3. Даны отношения: Rl (KP,Name,Adres)
   R2 (KP, KT, Cena), где KP – код поставщика, KT – код товара. На языке SQL определить адрес поставщиков, поставляющих товары ценой более 1 млн. рублей.
4. Даны отношения: STUDENTS (FIO,GRUPPA)
   SESSION (FIO,PREDM,OCENCA)
   На языке SQL выдать сведения о студентах заданной группы, получивших по курсу «Информатика» отличные оценки.
5. Даны отношения: OTDEL(NO,Name,Rukov)
   DOLGN(ND,NAZV)
   YKOMPL(NO,ND,KOL)
   На языке SQL выдать сведения об отделах и их руководителях, в которых имеются работники на должностях конструктора 1-й категории.
6. Дано отношение (Наименование_поставщика, Наименование_детали, Номер_Госта, Единица_измерения, Цена).
   На языке SQL изменить (увеличить) цену детали Д501 на 7 тысяч рублей.
7. Дано отношение: ПОСТАВКИ (Номер_партии_товара, Название_товара, Цена, Учетный_номер, Количество). На языке SQL произвести группировку товаров по наименованию товаров с учетом суммарного количества.
8. Даны отношения: Аптека (НА, Адрес, Телефон)
   Препараты (НП, Наименование, Вес, Цвет)
   Наличие (НА, НП, Количество)
   На языке SQL выполнить следующие задачи:

   8.1. Выдать сведения об аптеках, в которых имеется аспирин.

8.2. Выдать названия лекарств, имеющихся в аптеке А201.

8.3. Выдать общее количество лекарств, имеющихся в аптеке А201.

8.4. Выдать перечень препаратов и их суммарное количество по каждому наименованию в аптеке А201.

8.5. Указать номера аптек и их адреса, в которых имеется препарат «Винибис» и среднею цену данного препарата,

8.6. Получить на языке SQL сведения об аптеках, в которых имеются лекарства, указанные клиентом.

9. Даны отношения: Поставщики(П), Детали(Д) и Поставки(ПД)

П (НП, ФИО, Состояние, Город)
Д (НД, Название, Цвет, Вес, Город)
ПД (НП, НД, Количество)

9.1. Измените цвет всех красных деталей на зеленый.

9.2. Удалите все детали, которые отсутствуют в поставках.

9.3. Увеличьте размер поставки на 10% для тех деталей, которые поставляются поставщиком П1.

9.4. Удалите все детали, поставляемые из города Самары.

9.5. Вставьте в таблицу П нового поставщика. Его ФИО и город - Котов А.В. и Уфа соответственно, а состояние еще неизвестно.

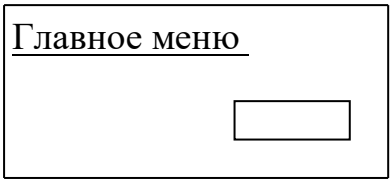
9.6. Добавьте 15 к состоянию всех поставщиков, состояние которых меньше, чем состояние поставщика П5.

9.7. Установите нулевой объем поставок для всех поставщиков из Уфы.

СУБД FOXPRO 2.0

СУБД FoxPro 2.0 пока все еще является популярной системой реляционного типа. Язык программирования, используемый системой, является основой для целого семейства так называемых dBASE-подобных СУБД, родоначальником которого является СУБД dBASEII, предложенная фирмой Ashton-Tate.

7.1. Системный интерфейс FoxPro, главное меню



Главное меню включает пункты:
SYSTEM, FILE, EDIT, DATABASE, RECORD, PROGRAM, WINDOWS.

SYSTEM – меню. Здесь реализованы средства доступа к файлам, HELP, а также «настольная оргтехника» – календарь, калькулятор и т.д.

FILE – меню. Средства управления – открытие, закрытие, создание файлов.

EDIT –меню. Работа с текстовым редактором.

Клавиши управления в редакторе. Традиционные клавиши перемещения : →, ←, ↑, ↓, **Pg Dn, Pg Up, Del, Ins** и другие:
Ctrl → / ← - на слово вправо/влево.
Home / End - на начало / конец строки.
Ctrl – Home / End - начало / конец текста.
Ctrl – W - сохранение отредактированного текста.
Esc или Ctrl – Q – выход без сохранения.
Выделение фрагментов текста:
Shift – → / ← - символа справа / слева.
Shift – ↑ / ↓ - строки.
Shift – Ctrl – → / ← - до конца / начала слова.
Shift – Ctrl – End / Home – до конца / начала текста.
Ctrl – A - всего текста.
Ctrl – X - удаление выделенного текста в буфер.
Ctrl – C - в буфер без удаления.
Ctrl – V - взятие из буфера.
Ctrl – U - отказ от предидущей операции.
Ctrl – R – возврат после отказа.
Ctrl –F / Ctrl – G - поиск вхождений.
Ctrl – E – замена вхождений.
DATABASE - меню. Работа с базой данных – модификация, редактирование, добавление записей, просмотр и т.д.
RECORD – меню. Действия по обработке записей БД.
PROGRAM – меню. Работа с командными файлами.
WINDOWS – меню. Работа с окнами.

7.2. Архитектура СУБД FoxPro 2.0

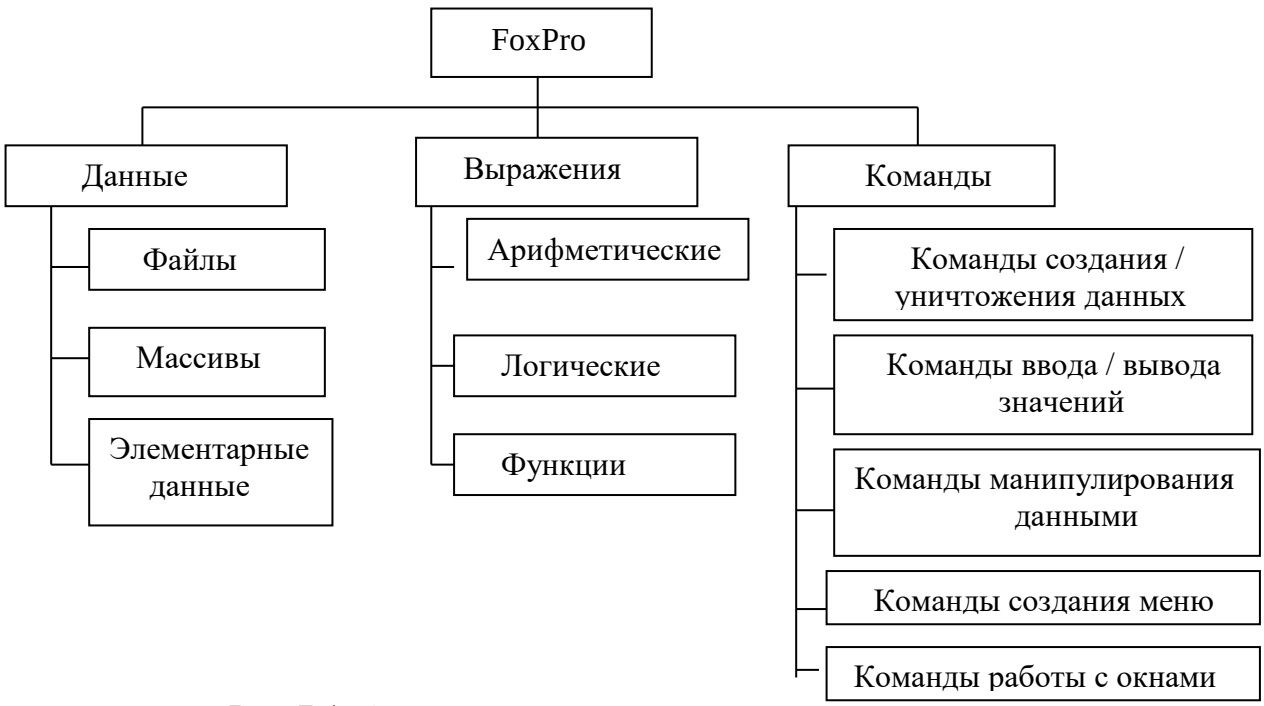


Рис.7.1. Архитектура FoxPro

Файлы могут быть:
- (.dbf) – БД.
- (.fpt) – файл примечаний.
- (.idx) – индексный файл.
- (.prg) – командный, программный файл.
- (.mem) – файл для сохранения временных переменных.
- (.exp) – откомпилированный командный файл.

К элементарным данным относятся:
- константы;
- переменные (до 256);
- поля записей.

Арифметические операции: +, –, *, /, ∧ или ** (степень), % (остаток от деления), () (скобки).

Отношения: <, >, =, ≠, ≥, ≤, $ (частичное совпадение), == (полное тождество).

Операции конкатенации: + (сцепление строк).

Логические операции: .NOT., .AND., .OR. .

Функции: математические функции; строковые функции; функции работы с датами; функции преобразования типов данных; функции проверки файлов и дисков; функции работы с мышью; клавишные функции; технические функции; функции времени; функции анализа условия; Функции анализа типа и наличия данных; финансовые функции; функции подстановки.

Общее описание.

Число записей – до 1 млрд.

Размер записи (в байтах) – до 4000.

Число полей в записи – до 255.

Число одновременно открытых баз – до 25.

Типы и размеры полей (в байтах).

Символьное поле – до 254.

Числовое поле – до 20.

Поле дат – 8.

Логическое поле – 1.

7.3. Основные команды FoxPro 2.0

Создание файлов: CREATE, INDEX, MODIFY COMMAND.

Добавление данных: APPEND, INSERT.

Открытие / закрытие БД: USE, CLOSE, CLEAR ALL, QUIT.

Удаление данных:

ERASE – удаление файла.

ZAP – удаление записей.

DELETE – пометка записей к удалению.

PACK – физическое удаление помеченных записей.

Комментарии: * - комментарий.

Построчный комментарий &&
Выдача данных: ?, ??, DISPLAY, LIST, BROWSE, @...SAY
Изменение данных: BROWSE, CHANGE, EDIT
Команды присваивания: <переменная> = <выражения>
STORE <выражение> TO <список переменных>
Команды управления: CLEAR, SET
Команды программирования:
DO
DO WHILE - ENDDO
FOR i = n TO m - ENDFOR
SCAN-ENDSCAN
DO CASE - ENDCASE
IF - ENDIF
MODIFY COMMAND
PROCEDURE – RETURN

7.4. Создание и редактирование БД

Для создания БД в интерактивном режиме используется команда CREATE [<имя файла>]. Например необходимо создать базу данных по составу кадров какого - либо предприятия. Пусть отношение имеет вид:
KADR (FIO, OTD, DATA, OKL), структура таблицы представлена на рис.7.2

KADR (кадры)

FIO (имя)	OTD (отдел)	DATA	OKL (оклад)
20 байт	10 байт	8 байт	6 байт
Петров	САПР	15/04/95	500

Рис.7.2. Структура таблицы

Чтобы создать такую таблицу, вводится команда CREATE .
Система выдает ответ: ‘Введите имя БД’. В ответ на это сообщение пользователь должен ввести имя таблицы, например,’KADR’. Далее производится ввод полей, типы данных и размеры, т.е. задается структура базы данных.

Filed name	type	width	dec
FIO	char/text	20	
OTD	char/text	10	
DATA	date	8	
OKL	numeric	6	2

После ввода структуры система сразу предложит вводить данные. Для добавления записей в конец уже существующей БД используется команда

APPEND [BLANK]

Для добавления новых записей в середину файла используется команда INSERT [BLANK] [BEFORE]

Синтаксис команды СУБД.

НАЗВАНИЕ [<границы>] [<список выражений>] [FOR<условие>] [WHILE<vcnoBHe>]

Название – имя команды.

Границы:

ALL - весь диапазон базы данных

REST - следующий диапазон базы данных ниже курсора

NEXT N - следующие N записей

RECORD N – N–я запись.

Примеры:

LIST ALL a,b,c FOR a<b WHILE c =100

LIST ALL fam, klass FOR klass="10A"

Просмотр данных:

LIST, DISPLAY, BROWSE, CHANGE.

Переходы по БД:

GO TOP

GO BUTTON

GO N

SKIP

Поиск данных и локализация. Фильтрация данных:

SET FILTER T0<условие>

SET FILTER TO FAM = “Ан”

Поиск:

LOCATE FOR <условие> [WHILE<условие>]

CONTINUE - продолжение поиска.

Пример: В базе данных успеваемость отыскать учеников, имеющих по Физике отличные оценки.

USE YSPEV

LOCATE FOR OCH = 5 .AND. PRD = "Физика"

? FAM Крылов

CONTINUE

? FAM Иванова

Индексирование БД. Индексирование базы данных производится для ускорения поиска информации.

INDEX ON <выражение> TO <IDX-файл> [COMPACT]

TAG <имя тега> [OF <CDX–файл>][FOR<условие>]

Пример индексирования БД YCHEN в порядке возрастания фамилий (рис.7.3).
USE YCHEN
INDEX ON FAM TO POFAM COMPACT

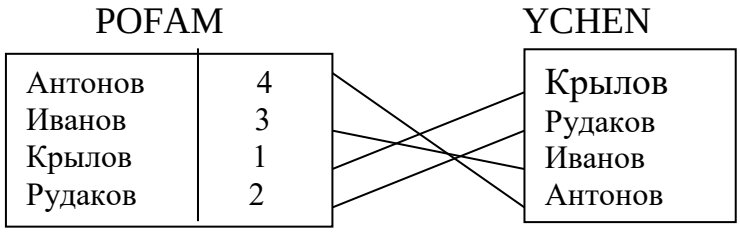


Рис. 7.3. Табл. POFAM-индексная, YCEN-исходная

При индексации создается индексный файл, позволяющий значительно ускорить поиск информации.
? INDEX ON FAM TAG fam
Ускоренный поиск осуществляется командой SEEK <выражение>

Работа с несколькими БД:
SELECT a
USE YCHEN
SELECT b
USE YSPEV
LOCATE FOR PREDM = "Физика" AND OCENKA=2
SELECT a
LOCATE FOR FAM = b. FAM
? FAM, ADRES

Управление индексами:
SET ORDER TO [<N><IDX->]
SET ORDER TO
USE kaders
INDEX ON FIO TO IDXFIO
INDEX ON OTD2 TO IDXOTD
……
SET ORDER TO 1
……
SET ORDER TO 2
Если индекс уже создан, то используется команда
USE SOTR INDEX SOTRIDX
Для установки индекса команда имеет вид:
SET INDEX TO SOTRIDX, SOTROTD2
Главным индексным файлом является файл открытый самым первым.

7.5. Команды просмотра и редактирования записей

BROWSE-ОКНО. С помощью данной команды можно просматривать записи в табличном виде, редактировать, дополнять и помечать к удалению. Она является наиболее мощным и удобным инструментом доступа к данным БД.

BROWSE
[FIELDS <поля>] [FOR / WHEN <условие>]
[TITLE <выражение>] [KEY <выражение 1> [, <выражение 2>]]
[LEDIT / REDIT] [WINDOW <окно>]
FIELDS может сопровождаться ключами:
[: КЛЮЧ] Ключи = R, выражение N, V, P, H, B, W.
R - разрешает только просмотр.
Выражение N - видимый размер поля.
H - указания собственного заголовка поля.
P - задание формата.

CHANGE –окно. Команда редактирования полей базы данных.

CHANGE [<границы>] [FOR<условие>]
[WHILE <условие>] [FIELDS <поля>] [опции]
Опции соответствуют BROWSE.
KEY - ограничение действия команды диапазоном ключевого выражения: <выражение 1>и <выражение 2> активного индексного файла.

7.6. Создание командных файлов

MODIFY COMMAND [MODI COMM] <имя файла> – создание командного файла
DO <имя файла> - запуск файла на выполнение.

Пример командного файла

```
SET TALK OFF
SET DATA GERMAN
CLEAR
USE YCHEN INDEX POFAM
BROWSE TITLE "СПИСОК УЧЕНИКОВ"
USE
```

К основным операторам языка FoxPro можно отнести:
1) команды ввода / вывода;
2) команды управления;
3) команды организации связи с программами.

Команды ввода / вывода.
?, ??, TEXT...ENDTEXT, ACCEPT, WAIT, INPUT
@...SAY...GET, READ**,** \, \\ .
Команды: ?, ?? - выдают на экран значение указанного выражения.
Например: ? "Это база данных" —> Это база данных
? 5+5 -> 10
? 5*5 -> 25

? - значение печатается в новой строке.

?? - значение печатается в текущей строке.

Конструкция, обеспечивающая вывод на экран блока текста:
TEXT
<TEXT>
ENDTEXT

Команда ввода INPUT [<подсказка>] TO <переменная>
Например: FIO = ‘ ‘
INPUT ‘Введите имя’ TO FIO
‘Иванов’

Команда ACCEPT [<подсказка>] TO <имя переменной>
Предназначена для ввода с клавиатуры и помещение введенной информации в указанную переменную. Данные вводятся в виде строки знаков. Используется для ввода подсказок.
Например:
ACCEPT "Введите имя поля базы данных" TO Answer ? ? & Answer

Команда WAIT [<подсказка>] [TO <имя переменой>]
Выдает на экран описанную подсказку и приостанавливает выполнение программы до нажатия любой клавиши.
Например:
WAIT "Введите начальное значение переменной X"
STORE X TO 10

FoxPro позволяет осуществлять ввод / вывод с любой позиции дисплея по ее координатам. Строки экрана нумеруются от 0 до 23, а позиции от нуля до 79. Реализуется командой @.

Команда @ <строка, позиция> [SAY <выражение> [PICTURE формат>]]

[GET <выражение> [PICTURE <формат>]]

Она обеспечивает вывод в указанную позицию экрана значение выражения или вводит подготовленные пользователем данные в указанном формате координат, либо очищает экран.
Например:

```
X =" "
@ 7, 20 SAY "Введите один из возможных ответов"
@ 9,20 SAY "П - продолжить работу"
@ 11,20 SAY "З - закончить работу"
@ 12,20 GET X
READ
```

SAY - выдает подготовленные пользователем данные на экран или принтер.
GET - выдает подготовленные данные на экран для редактирования.

Команда @ <строка, позиция> [CLEAR]

стирает правую часть указанной строки после указанной позиции.

Команда @ <строка 1, позиция 1> TO <строка 2, позиция 2> [DOUBLE] рисует блок с левым верхним углом в <строке 1, позиции 1> и левым нижним углом в <строке 2, позиции 2>.
Если используется опция DOUBLE, то линии сдвоенные.

Шаблоны опции PICTURE. К символам шаблона относятся:
9 - цифры и знаки для числовых полей и цифры для символьных полей.
- только цифры, знаки и пробелы.
A - только латинские буквы.
L - для логических данных.
N - буквы и цифры (буквы латинские).
X – любой знак.
I - любой знак, но строчные буквы переводятся в прописные.
. - указывает на положение десятичной точки.
, – служит для отделения тысяч.

В шаблоне могут присутствовать и любые другие символы:

```
PICTURE "999 рублей 99 копеек"
PICTURE "Фамилия: AAAAAA"
```

Пример:

```
STORE 2507.6451 TO NUM
@ 5,41 SAY NUM PICTURE "####.####"    2507.6451
@ 7,20 SAY "abcd" PICTURE "!!!!"      A B C D
```

Существуют еще и функциональные знаки:
S(n) - ограничивает ширину поля просмотра до п символов.

(-заключает отрицательные числа в скобки.
R - литеры в шаблонах не являются частью вводимых данных.
Z - отображает нуль как пробел.
D - используется американский формат даты.
E - европейский формат даты.
B - числовые данные выводятся с левой границы поля.
Например, надо вывести таблицу:

СВЕДЕНИЯ О СОТРУДНИКАХ

N	ФИО	Должность	ГР
4	20	10	8

Программа будет иметь следующий вид:

```
CLEAR
Cherta = replicate ('_', 46)
@ 1,10 SAY "СВЕДЕНИЯ О СОТРУДНИКАХ"
@ 2, 1 SAY Cherta
@ 3, 1 SAY "| N |          ФИО          | Должность|   ГР    |"
@ 4, 1 SAY Cherta
```

7.7. Команды управления

Условный оператор IF:

```
IF <условие>
    Команды
ELSE
    Команды
ENDIF
```

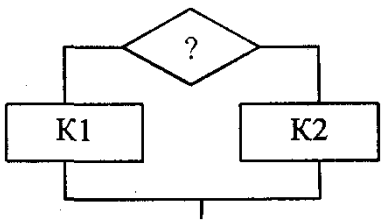


Рис.7.4.Блок-схема оператора IF

Пример:

$$Y=\begin{cases} A, & X<0 \\ B, & X\geq 0 \end{cases}$$

```
IF X < 0
  STORE  A  TO  Y
ELSE X >= 0
  STORE  B TO  Y
ENDIF
```

Оператор IF может применятся и для выбора типа "один из множества". Например, если возраст работающих разбит на группы: до 18 лет, 18-28 лет, 29 - 60 лет и старше 60 лет, то

```
IF Vozrast<18
<команды> ELSE
                IF Vozrast>=18.AND.<= 28
                        <команды>
                    ELSE
```

```
IF Vozrast > 28. AND. <=60
        <команды>
          ELSE
            <команды>
      ENDIF
    ENDIF
  ENDIF
```

Оператор DO CASE:

```
DO CASE
    CASE <условие>
        <команды>
    CASE <условие>
        <команды>
  ……..
    [OTHERWISE]
    <команды>
ENDCASE
```

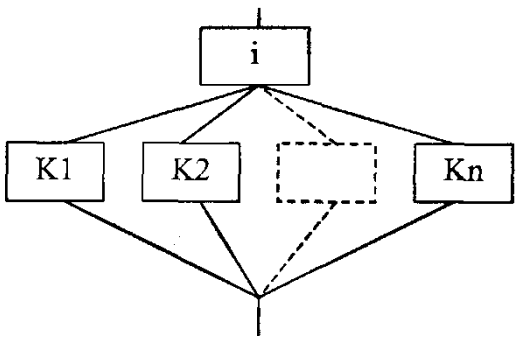


Рис.7.5. Блок–схема оператора выбора DO CASE

Оператор выполняет проверку условий, заданных в операторах CASE. Они просматриваются последовательно сверху вниз до истинного условия.

```
DO CASE
    CASE Vozrast< 18
        <команды>
    CASE Vozrast > = 18 .AND. Vozrast < = 28
        <команды>
    CASE Vozrast >28 .AND. Vozrast < = 60
        <команды>
    OTHERWISE    <команды>
ENDCASE
```

7.8. Циклы в FoxPro

Циклы с условием: DO WHILE <условие>

<команды>

ENDDO

Организуется циклическое выполнение группы команд пока <условие> истинно. В цикле могут быть использованы операторы LOOP, EXIT. LOOP -обеспечивает внеочередную проверку условия цикла и передает управление на оператор следующий за ENDDO, если условие не выполняется. Второй оператор обеспечивает безусловный выход из цикла.

Цикл с параметром:

```
FOR<переменные>=<выражение Nl>TO<выражение N2>[STEP выражение N3>]
        <команды>
    ENDFOR
```

Цикл сканирования БД:

SCAN [<границы>] [FOR / WHILE <условие>]

<команды>
ENDSCAN

При отсутствии границ и условий сканируется вся БД.

Примеры:

1)USE kadr
LOCATE FOR NAME ='П'
DO WHILE .NOT. EOF()
<обработка записей>
CONTINUE
ENDDO

2) USE kadr
SCAN FOR NAME='П'
<обработка записей>
ENDSCAN

3)USE kadr INDEX kadrfio
SEEK 'П'
DO WHILE FIO = ‘П’
<обработка записей>
SKIP
ENDDO

4) USE kadr INDEX kadrfio
SEEK 'П'
SCAN WHILE FIO=’П’
<обработка записей>
ENDSCAN

Пример 7.1. Нахождение факториала.

```
STORE 1 TO F
STORE 1 TO K
STORE 1 TO N
  DO WHILE K<=N
    F=F*K
    K-K+1
  ENDDO
? F
```

K	1	2	3	…	10
F	1	1*2	1*2*3	…	1*2*…*10

Пример 7.2. Вычисление ряда до 100 или с точностью до 0.001

$$S=1/1^3+1/2^3+1/3^3+....+1/N^3$$

```
STORE  1  TO  i,A
STORE  1 TO  S
DO WHILE i <= 100
  A = 1 / (i * * 3)
  S=S+A
  IF A < 0.001
    EXIT
  ENDIF
  i = i + 1
ENDDO
?  S
```

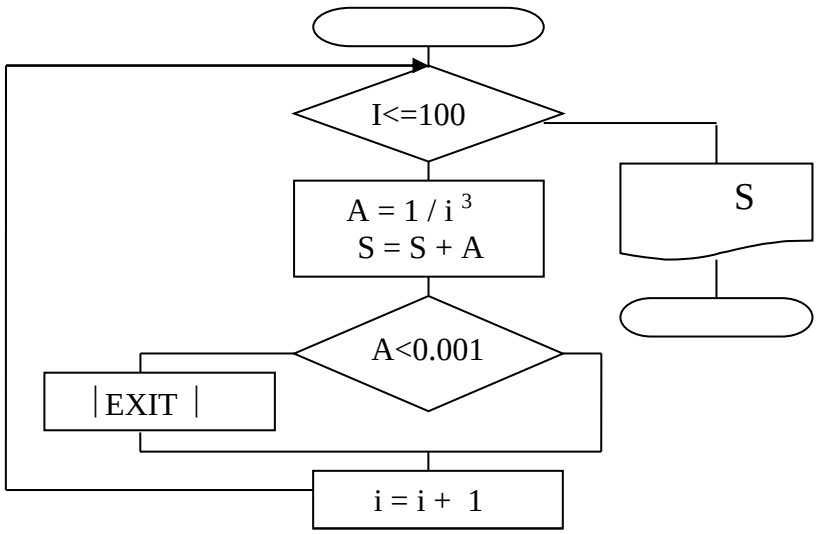


Рис.7.6. Блок-схема вычисления ряда

Пример 7.3. $Y = X^{100}$

```
STORE  0   TO   i
STORE  1   TO  Y
    DO  WHILE    i <= 100
        Y = Y * X
        i = i + 1
   ENDDO
?  Y
```

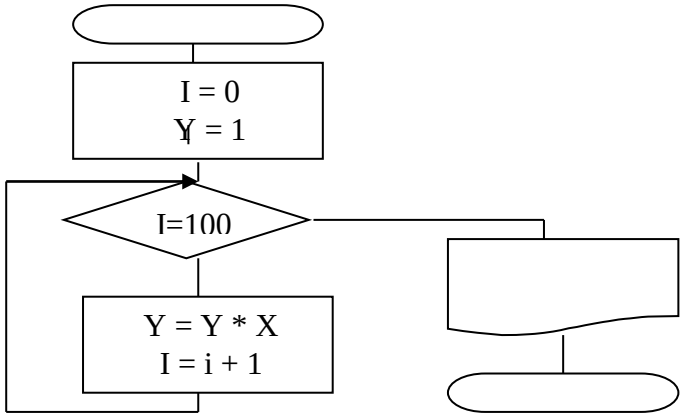


Рис. 7.7. Блок-схема вычисления $Y = X^{100}$

Пример 7.4. Вычислить ln2 с заданной точностью.

$\ln 2 = 1 - 1/2 + 1/3 - 1/4 + \ldots \pm 1/N$ с точностью до ε

```
 @  1,1 GET   EPS   PICTURE   ‘9.9999’
  READ
  STORE   1   TO   A
  STORE   1   TO   N
  STORE   1   TO   S
     DO  WHILE   A > EPS
        N = N + 1
        A = 1 / N
        R = N / 2 – INT (N / 2 )
        IF   R  >  0
           S = S + A
         ELSE
           S = S – A
        ENDIF
      ENDDO
?  S
```

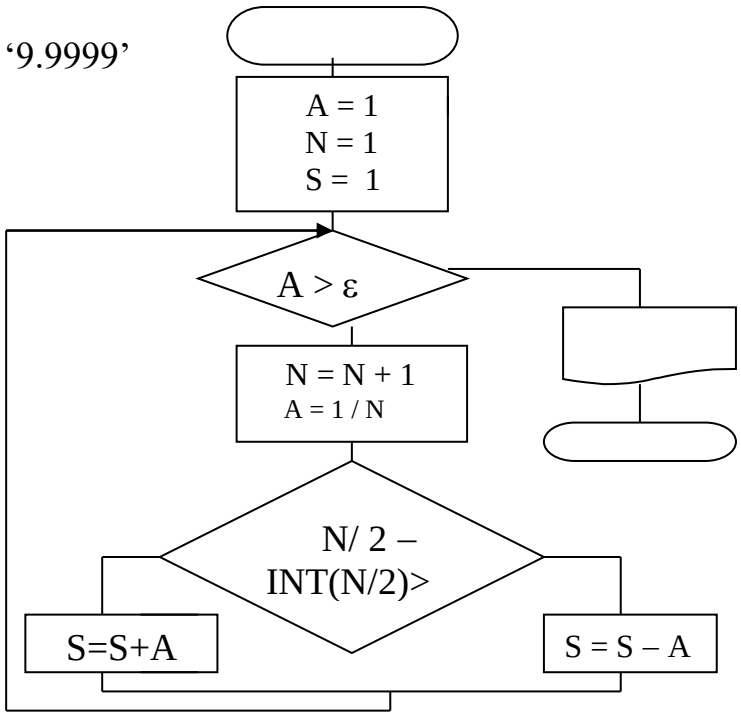


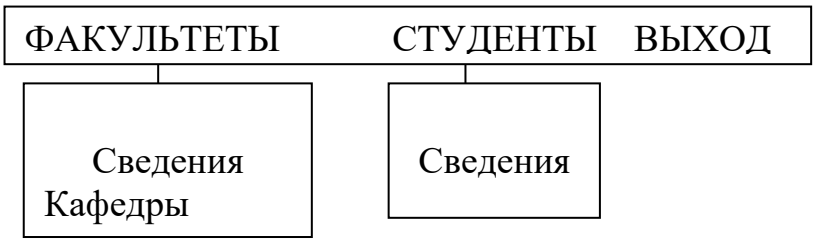
Рис.7.8. Блок–схема вычисления ln2

Практические задания

1. Запрограммировать на FoxPro следующее выражение

$$Y = \begin{cases} 2x + 5, & x<0 \\ 3 - 4x, & 0<=x<=2 \\ 3 + 20x, & x>2 \end{cases}$$

2. На языке FoxPro организовать dBASE-меню



3. Построить LIGHTBAR меню с пунктами : Группы, Студенты, Кафедры, Преподаватели.

4. Построить dBASE – меню вида

5. Имеется таблица:

УСПЕВАЕМОСТЬ (ФИО_студента, Группа, Дисциплина, Оценка).

На языке FoxPro выдать сведения о студентах заданной группы, получивших по курсу «Информатика» отличные оценки.

6. Составить в среде FoxPro двухуровневое меню, где бы по горизонтали были пункты: Отделы, Сотрудники, Выход, а по вертикали на пункт «Отделы» активировалось бы вертикальное меню с пунктами: Сведения, Работы.

7. Даны отношения: OTDEL (NO, Name, Rukov)
DOLGN (ND, NAZV)
YKOMPL (NO, ND, KOL)

На языке FoxPro, выполнить следующие задачи:

7.1. Распечатать сведения об отделах в алфавитном порядке.

7.2. Выдать сведения об отделах и их руководителях, в которых имеются работники на должностях конструктора 1-й категории.

7.3. Определить общее количество программистов во всех отделах.

7.4. Выдать список отделов в форматированном виде.

8. Дано отношение: ПОСТАВКИ (Номер_клиента, ФИО, Адрес, Номер_партии_товара, Название_товара, Цена, Учетный_номер, Количество). Получить на языке FoxPro сведения о клиентах, находящихся в Самаре, в форматированном виде .

9. Составить в среде FoxPro двухуровневое меню, где бы по горизонтали были пункты: Фирмы, Клиенты, Товары, Выход, а по вертикали на пункт «Фирмы» активировалось бы вертикальное меню с пунктами: Сведения, Работы, Отделы.

10. Дана таблица: ВЕДОМОСТЬ (Номер_группы, Специальность, ФИО_студента, Номер_зачетной_книжки, Дисциплина, Оценка).

На FoxPro выдать список задолжников в виде таблицы с полями:

N, ФИО, Номер зачетной книжки, Дисциплина.

11. Дано отношение: НАЛИЧИЕ_ЛЕКАРСТВ_В_АПТЕКАХ (Номер_аптеки, Адрес, Телефон, Номер_лекарства, Наименование, Стоимость, Вес_упаковки, Количество_лекарства_в_данной_аптеке). На языке FoxPro написать программу выдачи перечня аптек, в которых имеется указанное лекарство в количестве более 100 упаковок.