СООБЩЕНИЯ
ОБЪЕДИНЕННОГО
ИНСТИТУТА
ЯДЕРНЫХ
ИССЛЕДОВАНИЙ
ДУБНА

10 - 8799

Д.Д.Арнаудов, З.И.Коженкова

ЯЗЫК ЗАПРОСА ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ ОИЯИ

1975

10 - 8799

Д.Д.Арнаудов, З.И.Коженкова

ЯЗЫК ЗАПРОСА ИНФОРМАЦИОННО-ПОИСКОВОЙ СИСТЕМЫ ОИЯИ

Язык запроса является своего рода посредником между пользователем и системой автоматического поиска и хранения информации. Он должен позволять пользователю реализовывать все проектные возможности, заложенные в структурно-функциональной организации массивов. Язык запроса будет обеднен, если в нем не реализованы все возможности, заложенные структурой массива и стратегией поиска, или в том случае, когда пользователь неправильно понимает или неумело использует язык. В связи с этим возникает вопрос структуры языка, при решении которого существуют две крайние точки зрения[1]. Язык получает гибкость и выразительную силу при наличии обогащенного словаря и контекстно-чувствительного синтаксиса. Это, конечно, приводит к неоднозначности в языке, что легко иллюстрируется на примере естественных языков.

Другая крайность находит свое выражение в языках, подобных машинному коду или языку большинства трансляторов, в языках с контекстно-свободным синтаксисом и очень ограниченным словарем.

Где-то посредине в этой классификации находятся формализованные дескрипторные языки различного вида.

В существующих ИПС используется вся гамма разновидностей языков запроса. Так, например, в системе "SMART"[2] в качестве языка запроса и языка описания документов используется естественный английский язык. Но даже здесь при осуществлении конкретной стратегии поиска для раскрытия критерия смыслового соответствия необходимы методы последовательной нормализации языка, которые реализуются при помощи соответствующих словарей (словарь отрицаний, синонимов, словосочетаний и т.п.). При этом словари не полностью устраняют неточности языка, но уменьшают влияние многих нарушений норм, используя

соответствующие алгоритмы формирования словаря. В системе "SMART", например, соответствие между словом и единичным понятием может получить больший вес, чем соответствие между словом и множеством понятий, так как первое, по-видимому, представляет единственное значение для этого слова, в то время как последнее предполагает неоднозначность. Вообще, использование естественного языка, как в качестве языка запроса, так и в качестве языка для представления содержания документа в современных ИПС, недостаточно эффективно из-за ограниченности памяти и из-за невозможности пока устранить неоднозначности в естественном языке.

Интерес представляет язык запроса и язык описания содержания документа в системе "STAIRS" /3/. Здесь содержание описано естественным языком, а язык запроса строго формализованный (он основан на применении булевских операторов над отдельными словами/словосочетаниями/). Из-за большого объема хранимой информации в памяти ЭВМ при большом поисковом массиве (больше 100 000 документов) характеристики системы резко ухудшаются.

Самым перспективным на сегодняшнем этапе (и в недалеком будущем) является использование формализованных языков, как в качестве языка запроса, так и для описания документов. Формализация естественных языков может производиться различными путями. Под формализованным естественным языком мы понимаем язык, в котором фиксирован в виде тезауруса состав четко определенных элементарных терминов (дескрипторов) и фиксирован набор четко определенных смысловых отношений, которыми могут связываться дескрипторы между собой, образуя сложные понятия.

Состав дескрипторов определяется терминологией той конкретной

тематики, для которой предназначается ИПС. Состав смысловых связей определяется структурой вопросов и ответов, которые должны циркулировать в ИПС, и связан неразрывно с назначением ИПС. Так, например, при поиске научной медицинской информации[4] имеется восемь различных смысловых связей между дескрипторами ("место","причина","следствие" и т.п.). Конечно, выбор необходимого и достаточного набора подобных смысловых связей не может быть произведен однозначно. Он делается в основном интуитивно на базе анализа потребностей в информации для определенной группы заказчиков. При этом, вероятно, существует некоторый оптимальный состав связей и отклонение от него, как в сторону сокращения числа связей, так и в сторону их увеличения приводит к снижению эффективности ИПС.

Дескрипторные языки, где дескрипторы могут быть связаны между собой различными смысловыми связями, известны как "языки с грамматикой"[4]. Но основным недостатком подобных языков является то, что при использовании смысловых отношений возможно снижение полноты выдачи. Кроме того, введение этих связей значительно усложняет процедуру индексирования, информационный поиск и делает их более дорогостоящими. Усложнение информационного языка введением в него грамматических средств оправдывает себя при необходимости глубокого представления поискового образа документа и запроса для фактографических поисковых систем.

В связи с этим существуют по крайней мере три причины, которые являются существенными и вследствие которых неприемлем выбор дескрипторного языка с грамматикой, как в качестве языка запроса, так и в качестве языка описания документов в ИПС ОИЯИ:

I. ИПС ОИЯИ является системой документального типа.

2. Средняя глубина индексирования документов – десять дескрипторов.

3. Основная часть входного потока в систему формируется в ИНИС (Вена) и документы воспринимаются ИПС ОИЯИ уже в формализованном виде (они уже индексированы дескрипторами без смысловых связей).

В качестве языка описания документов и языка запросов в ИПС ОИЯИ выбирается дескрипторный язык без грамматики.

Для отражения более полного соответствия между языком запроса, с одной стороны, и структурно-функциональной организацией поиска, с другой стороны, в языке запроса можно использовать комбинацию дескрипторов с операторами булевской алгебры ("и","или","нет"). Кроме того, возможно формирование запросов с помощью характеристик, не являющихся дескрипторами, соединенных знаками ($>, <, \geq, \leq, \neq, =$, и т. д.).

Для более правильного формирования запросов потребителем ИПС может выступать в качестве автоматического словаря, что даст возможность предоставить пользователю информацию о более "узких" и более "широких" дескрипторах, о частоте использования дескрипторов, о "весе" дескриптора и т.п.

Остановимся более подробно на булевской форме запроса.

Как известно, существуют три типа запросов:

а) Простой запрос. Это набор дескрипторов (формально это есть набор дескрипторов, связанных булевским оператором "и").

б) Запрос с рангом. Обычно это запрос, состоящий из терминов, не являющихся дескрипторами, например: найти все документы, где "цена" равняется I8,00.

с) Булевский запрос. Состоит в общем случае из двух прежних, комбинированных знаками $\wedge, \vee, 7$ (знаками конъюнкции, дизъюнкции, отрицания).

Группу запросов булевского типа, одновременно обрабатываемых, будем рассматривать в качестве запроса системы.

Пусть имеем некоторую минимизированную дизъюнктивную форму запросов: $(K_1 \wedge 7\ K_2 \wedge \ldots \wedge K)\ \vee (\ldots) \ldots \vee (\ldots)$ /5/

Такой запрос легко разбивается на некоторое число конъюнкций и рассматривается как состоящий из некоторого количества отдельных запросов.

Легко показать, что для множества дескрипторов в данной конъюнкции число обрабатываемых записей во время поиска можно сделать наименьшим.

Пусть $L_{i,j}$ означает K_i -список с начальным адресом $a_{i,j}$; пусть L_i - это множество всех h_i K_i -списков, где h_i - число различных списков (K_i - списков).

Мы имеем $L_i = \bigcup_{j=1}^{h_i} L_{i,j}$, $i = 1,2 \ldots r$. Можно отметить, что каждый L_i список содержит в точности n_i записей. Множество S записей, удовлетворяющее конъюнкции $(K_1 \wedge K_2 \ldots K_r)$, фактически является пересечением L_i списков.

Следовательно, $S = \bigcap_{i=1}^{r} L_i = \bigcap_{i=1}^{r} \bigcup_{j=1}^{h_i} L_{i,j}$.

Дальше мы отмечаем, что S есть подмножество L_i для каждого i. Конечно, эти L_i могут иметь разные длины. Очевидно, гораздо труднее найти S для большого L_i. Чтобы минимизировать усилия при нахождении S , необходимо исследовать самый короткий L_i. Именно это позволяет иметь наименьшее подмножество обрабатываемых записей для каждой конъюнкции в запросе. В связи с этим и особенностя-

ми выбранной стратегии поиска[6] выбираем дизъюнктивную форму как самую удобную и эффективную для обработки запросов в ИПС ОИЯИ.

Рассмотрим подробнее обработку поступающих в систему запросов. Каждый запрос представляется в виде дизъюнктивной формы, например:

$$A \wedge 7B \wedge C \vee A \wedge D \wedge F \vee A \wedge K \vee \ldots A \wedge B \wedge 7C \wedge D \quad (I)$$

Необходимо сказать, что A,B,C и т.д. являются кодами соответствующих дескрипторов из дескрипторного словаря. Перед тем, как получить их, запрос формируется пользователем, использующим дескрипторный словарь системы. Форма сформулированного запроса следующая: abstract $\wedge$ radiation $\wedge 7$ even-event.

Этот запрос воспринимается входным блоком системы, и все дескрипторы получают некоторый эквивалент, соответствующий коду дескриптора, т.е. получается выражение (I).

После этого происходит минимизация данной булевской формы. Необходимо отметить, что никакие ограничения не накладываются на употребление оператора отрицания внутри любой конъюктивной формы (можно иметь, например, как одно, так и несколько отрицаний).

Цель алгоритма минимизации – определить полные классы булевских множеств и избежать повторной работы из-за эквивалентности классов и из-за образования пустых множеств.

На рисунке I показана картина трех булевских операторов:

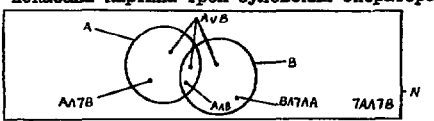

Рис.I.

N – общее множество документов,

A – класс документов, индексированных дескриптором A,

B – класс документов, индексированных дескриптором B,

$\vee$ – знак булевского оператора "или",

$\wedge$ – знак булевского оператора " и ",

$\neg$ – знак булевского оператора " не".

В алгоритме минимизации используются следующие булевские соотношения:

I. $a \wedge b = b \wedge a$	7. $\neg a \vee b = \neg(a \wedge \neg b)$
2. $a \wedge (b \wedge c) = (a \wedge b) \wedge c = a \wedge b \wedge c$	8. $a \vee \neg b = \neg(\neg a \wedge b)$
3. $a \vee b = b \vee a$	9. $a \wedge \neg b = \neg(\neg a \vee b)$
4. $a \wedge \neg a = 0$	I0. $a \vee (a \wedge b) = a$
5. $a \vee \neg a = 1$	II. $\neg a \wedge \neg b \ \ldots = (\ldots a\ b\ c \ldots$
6. $\neg a \vee \neg b = \neg(a \wedge b)$	I2. $a \wedge \neg b \vee a \wedge b = a$

Описание алгоритма восприятия и минимизации булевской формы запросов

Алгоритм восприятия запросов и минимизация булевской формы состоит из ряда программ, выполняющих следующие функции:

1. Ввод поступающих запросов.
2. Анализ отдельных запросов и замена имени дескриптора соответствующим цифровым кодом.
3. Минимизация дизъюнктивной формы запросов.

На вход системы поступает массив запросов ZAP ; причем каждый запрос, представленный в виде булевской дизъюнктивной формы, оканчивается точкой (·). В этой булевской функции переменной являются названия дескрипторов, которые связаны между собой следующими знаками логических операций: $\vee, \wedge, \neg$.

Каждый запрос может иметь следующий вид:

descr 1$\wedge$ descr 2$\wedge$ descr 3$\wedge\neg$ descr 4$\wedge$ descr 1$\vee$ descr 1$\wedge$ descr2$\wedge$ descr 20$\wedge$ descr 105$\wedge$ ... descr 7$\wedge$ descr 4$\wedge$ descr 1$\vee$... descr 8.

Входной массив запросов ZAP состоит из ряда таких запросов.

Структура массива ZAP показана на рисунке 2. Во входном массиве

запросов ZAP содержится информация о нескольких запросах. Информация одного (i) запроса будет содержаться в $n_i = entr\left[\frac{M_i}{80}+1\right]$ перфокартах, где M_i – количество символов (включая скобки и пробелы в названиях дескрипторов), при помощи которых был записан i-й запрос. Каждый запрос оканчивается точкой (•).

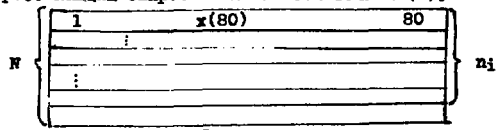

Рис.2. Структура входного массива запросов ZAP

Входной массив ZAP состоит из пакета $N=\sum_{i=1}^{NONZ} n_i$ перфокарт. Максимальное количество запросов (NONZ) в массиве ZAP не должно превышать 99. В алгоритме предусмотрена пакетная обработка группы запросов с максимальным общим числом дескрипторов – 1250. Задача состоит в том, чтобы минимизировать поступающие запросы, представленные при помощи булевских функций, и привести результаты минимизации запросов к виду, удобному для дальнейшей обработки. В процессе работы алгоритма формируются выходной массив KART , таблица дескрипторов запросов KORP-REC и рабочие таблицы APM-REC и VHOD. Структура таблицы KORP-REC показана ниже. Она создается при помощи дескрипторов, которые участвуют в образовании запросов пользователей и входит в массив ZAP .

	KORP(1)	KORP(2)	...	KORP(N_1)	...	KORP(29)	KORP(30)
	x	x	...	x	...	x	x
KURP	KORP(1,1)	KORP(1,2)	...	KORP(1,N2)	...	KORP(1,29)	KORP(1,30)
S ↓	⋮						
KURP(S,) S_1 ↓	KORP(S_1,1)	KORP(S_1,2)	...	KORP(S_1,N2)	...	KORP(S_1,29)	KORP(S_1,30)

I. Таблица дескрипторов KORP-REC .

Здесь S_1 -индекс дескриптора во входном массиве запросов ZAP (здесь единая индексация дескрипторов по всем запросам).

N_2 – индекс символов в дескрипторе. KURP(S_1) – сюда переписывается S_1 – й дескриптор из входного массива ZAP . KORP(S_1,N_2)– определяет N_2 – й символ S_1 -го дескриптора. Число символов KORP(S_1,N_2) в S_1-дескрипторе не может быть больше максимального размера дескриптора в дескрипторном словаре, т.е. N_2 не должно превышать 30. Количество дескрипторов KURP(S_1) должно быть равно количеству дескрипторов в запросах входного массива ZAP , т.е. S_1 не может превышать 1250. Структура таблицы запроса APN-REC показана ниже.

	APN(1)	APN(2)	. . .	APN(N_2)	. . .	APN(40)
	x(5)	x(5)		x(5)		x(5)
PAN(1)	APN(1,1)	APN(1, 2)	. . .	APN(1,N_2)	. . .	APN(1,40)
PAN(N_1)	APN(N_1,1)	APN(N_1,2)	. . .	APN(N_1,N_2)	. . .	APN(N_1,40)
PAN(50)	APN(50,1)	APN(50,2)	. . .	APN(50,N)	. . .	APN(50,40)

2. Таблица запроса APN-REC.

Она состоит из конъюнктивных групп PAN(N_1), здесь N_1 – индекс конъюнктивных групп в таблице APN-REC или индекс конъюнктивных групп в запросе. APN(N_1,N_2) – это N_2-й элемент в N_1-конъюнкции,т.е. здесь N_2 – индекс элементов в конъюнктивной группе PAN(N_1) .APN(N1,N2) равен либо символу "7", либо коду дескриптора. Один запрос входного массива ZAP может состоять из 50 конъюнкций, причем каждая конъюнкция может включать в себя 25 дескрипторов со своими отрицаниями. Таблица VHOD показана ниже. Структура выходного массива KART показана на рисунке 4.

	KD	IOT	IUS
S_1	9(5)	9	x

3. Таблица запросов VHOD .

Таблица VHOD заполняется при помощи дескрипторного словаря и входного массива запросов ZAP . Здесь используются следующие обозначения: S_1 – индекс строк таблицы VHOD ; он также является индексом дескрипторов запроса массива ZAP. Необходимо отметить, что число строк таблицы VHOD не должно превышать I250.

РД (S_1)– код S_1 дескриптора из массива запросов ZAP ; этот код берется из дескрипторного словаря по названию дескриптора; LOT(S_1)– если перед S_1-дескриптором в массиве запросов ZAP стоит знак отрицания (7), то в LOT(S_1) заносится "I", в противном случае –"0". LUS(S_1) – сюда записывается знак логической операции ($\vee$ или $\wedge$), который стоит после S_1-дескриптора в массиве запросов ZAP ; если после S_1-дескриптора в массиве запросов ZAP стоит точка (·), то в LUS (S_1) заносится SPACE(␣), что означает конец одного запроса в массиве ZAP.

Алгоритм состоит из шести этапов (блок-схема алгоритма показана на рисунке 3).

I этап.

Блок I. Вводится массив запросов ZAP ; в этом массиве общее количество дескрипторов должно быть не более I250, а количество запросов не должно превышать 99 .

Блок 2. Здесь происходит формирование таблицы дескрипторов KORP – REC и заполнение столбцов LOT и LUS таблицы VHOD . Для этого исследуется каждый элемент массива запросов ZAP на равенство "$\wedge$", "$\vee$", "7", "·" . Если элемент запроса равен "$\wedge$", "$\vee$", "·", то этот знак заносится в соответствующую строку в LUS (S_1) таблицы VHOD , если элемент запроса равен "7", то записывается I в соответствующую строку в LOT (S_1) таблицы VHOD . Однако если элемент запроса

не равен "V", "Λ", "7",".", тогда он заносится в рабочую таблицу дескрипторов KORP-REC , вернее, в KORP (S_1,N_2); здесь N_2 – индекс элемента в S_Iдескрипторе запроса, он не может быть больше 30, т.е. максимального размера дескриптора.

Блок 3. Здесь совершается заполнение столбца КД таблицы VHOD . Для этого используется дескрипторный словарь, откуда по названиям дескрипторов KURP (S_1) из KORP-REC находим коды этих дескрипторов, которые засылаются в соответствующие строки КД (S_1) таблицы VHOD . Производится одновременное заполнение строк КД (S_1), которым соответствуют одинаковые дискрипторы KURP (S_1).

II этап.

Блок 4. На этом этапе обрабатывается соответствующий запрос из таблицы VHOD и переписывается в некоторую таблицу APN-REC , состоящую из конъюнктивных групп PAN(N_1), каждая из которых состоит из элементов, где указывается N_1 – индекс конъюнктивной группы в запросе и N_1 – индекс элемента в N_1 -й конъюнкции . Каждая конъюнктивная группа PAN (N_1) представляет собой N_1 – ю строку таблицы APN-REC . Переписывание из таблицы VHOD в APN-REC происходит следующим образом. Если в LOT (S_1) стоит "I", тогда в APN (N_1,N_2) записывается знак "/", а в APN (N_1,N_2) где N_2= N_2+1 , записывается КД (S_1); в противном случае, КД (S_1) сразу записывается в элемент APN(N_1,N_2). Если в LUS(S_1) стоит знак "Λ", то переходим к рассмотрению следующей S_1= S_1 + 1 строки таблицы VHOD , и следующий полученный элемент APN(N_1,N_2), N_2 = N_2 +1 будет принадлежать к той же конъюнктивной группе PAN(N_1) , что и предыдущий элемент PAN(N , N).Если же в LUS(N_1) стоит знак "V", то рассматриваем следующую S_1=S_1+1 строку таблицы VHOD , и новый получен-

ный элемент APN (N_1 N) будет принадлежать новой конъюнктивной группе PAN (N_1) (здесь $N_1 = N_1+1$). Запрос кончается, когда в LUS (S_1) стоит SPASE. В этом случае получается таблица APN=REC, в которой отсутствуют знаки логических операций "$\vee$" и "$\wedge$" между элементами запроса, хотя и предполагается, что все элементы $APN(N_1,N_2)$ (кроме ";"), входящие в одну конъюнктивную группу $PAN(N_1)$, соединяются между собой знаками "$\wedge$", а все конъюнктивные группы $PAN(N_1)$ соединяются между собой знаком "$\vee$".

III этап. (первый этап минимизации).

Блок 5. На предыдущем этапе была получена таблица APN-REC, представляющая собой набор конъюнктивных групп $PAN(N_1)$ (их может быть до 40, а число элементов $APN(N_1,N_2)$ в них может быть до 50), без знаков логических операций "$\wedge$" и "$\vee$" в них. На этом этапе проводится минимизация внутри конъюнктивных групп $PAN(N_1)$, и здесь устраняются ложные конъюнктивные группы $PAN(N_1)$, т.е. такие группы, в которых присутствует один и тот же элемент, как в прямом виде, так и в инверсном. Короче говоря, на этом этапе к каждой конъюнктивной группе применяются следующие законы:

$$A \wedge A = A \qquad A \wedge \overline{A} = 0$$

IУ этап. (2-й этап минимизации)

Блок 6. На этом этапе начинается шаг минимизации. Здесь устраняются нулевые конъюнктивные группы ($PAN(N_1) = 0$) и единичные элементы в конъюнктивных группах ($APN(N_1,N_2)=1$). К каждой конъюнктивной группе применяется закон $A \wedge 1 = A$, а ко всем конъюнктивным группам $PAN(N_1)$ применяется закон $A \vee 0 = A$.

Блок 7. В конце этого этапа производится сортировка таблицы с целью расположения конъюнктивных групп $PAN(N_1)$ в порядке возраста-

ния количества элементов $APN(N_1,N_2)$ в этих группах. При сортировке применяется следующий закон ко всем конъюнктивным группам PAN(N): $A \vee B = B \vee A$. В таблице APN-REC содержится MAK элементов, а в каждой конъюнктивной группе $PAN(N_1)$ содержится $NOM(N_1)$ элементов $APN(N_1,N_2)$

У этап. (Третий этап минимизации).

Блок 8. На этом этапе рассматриваются попарно все конъюнктивные группы $PAN(N_1)$ и $PAN(N_4)$. При сравнении элементов этих групп $APN(N_1,N_2)$ и $APN(N_4,N_5)$ используются следующие законы алгебры логики: $A \wedge B = B \wedge A$; $A \vee A = A$; $A \vee A \vee B = A$; $A \wedge B \wedge C \vee A \wedge B \wedge \neg C = A \wedge B$; $A \wedge B \wedge C \vee A \wedge B \wedge \neg C \wedge D = A \wedge B \wedge C \vee A \wedge B \wedge D$.

Блок 9. После того, как будут рассмотрены попарно все конъюнктивные группы таблицы APN-REC , наступает конец одного шага минимизации данного запроса из массива ZAP .

Блок 10. Если число элементов $APN(N_1,N_2)$ в таблице APN-REC на данном шаге минимизации равно числу элементов $APN(N_1,N_2)$ в таблице APN-REC , полученной на предыдущем шаге минимизации, тогда совершается переход к следующему этапу работы алгоритма, т.е. к Блоку 11. В противном случае возвращаемся к Блоку 6.

УI этап.

Блок 11. После полной минимизации запроса идет обработка результатов минимизации для приведения их к виду, удобному для дальнейшей работы. Рассматриваются все конъюнктивные группы $PAN(N_1)$; если в них есть элементы с отрицанием, то происходит перестановка элементов $APN(N_1,N_2)$ внутри группы $PAN(N_1)$, на первые места ставятся элементы без отрицания, далее элементы $APN(N_1,N_2)$ со своими отрицаниями. После этого формируется массив KART . Структура этого массива показана на рисунке 4.

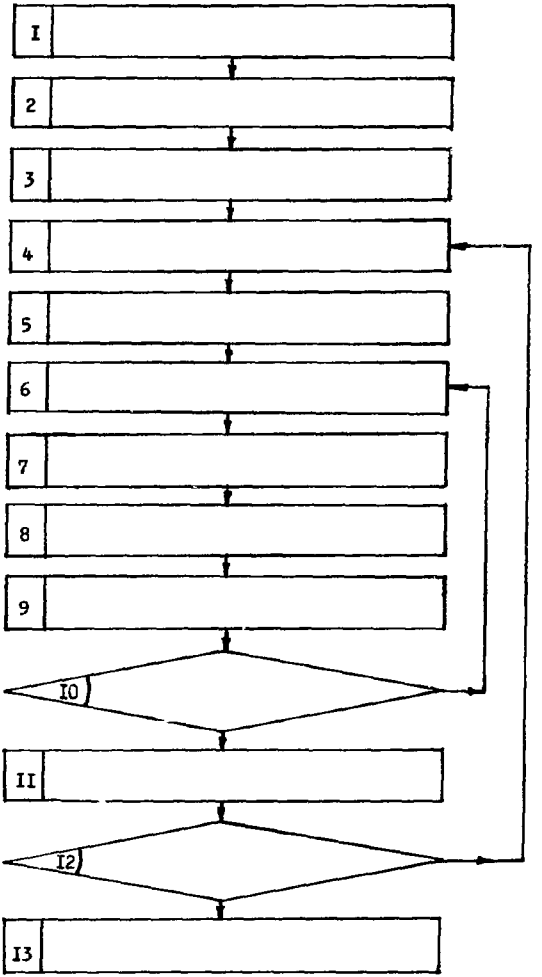

Рис.3.

S_3	NRZAP	NRGR	LOGS	COD
	9(2)	9	9	9(5)

Рис. 4. Структура выходного массива KART.

В массив KART заносятся результаты минимизации всех запросов; массив KART в таком виде будет передаваться для дальнейшей обработки. S_3 – индекс дескрипторов, записываемых в этот массив. Массив KART может содержать информацию о 1000 дескрипторах запросов. Для формирования массива KART рассматриваем все элементы таблицы APN-REC . В NRZAP(S_3) записывается номер обработанного (минимизированного) запроса. В NRGR(S_3) записывается номер рассматриваемой конъюнктивной группы PAN(N_1) в обрабатываемом запросе.

Если элемент APN(N_1,N_2) равен "/", тогда в LOGS(S_3) записывается "1", в противном случае туда заносится 0.

В СОД (S_3) записываются коды дескрипто[illegible]нимизированного запроса, т.е. сами элементы APN(N_1,N_2) , кроме APN(N_1,N_2)= / Блок I2. Если обработаны все запросы входного пакета, записанные в таблице VHOD , тогда (используя массив KART) переходим к Блоку I3 (дальнейшей обработке запросов). В противном случае приступаем к обработке следующего запроса из массива запросов ZAP , переписанного в таблицу VHOD , т.е. идем к Блоку4.

Для того, чтобы показать эффективность применения рассмотренного алгоритма минимизации в ИПС, был проведен машинный эксперимент. Результаты эксперимента представлены в таблице 4 и на рисунке 5.

Таблица 4

Результаты эксперимента

Кол-во запросов	Общее кол-во дескрипт.	Общее дескрипт. в явном виде	Кол-во дескрипт. после мин.	К-во дескрипт. в явном виде после мин.	Время ЦП	Астр. время (с)	Относ. время
I	25	I8	24	I8	I.24	4	4
2	50	39	4I	34	2.263	5	2.5
3	75	63	58	50	3.299	6	2
4	I00	82	8I	70	4.584	7	I.75
5	I25	I07	98	87	5.436	8	I.6
6	I50	I24	I2I	I06	7.03I	9	I.5
8	200	I66	I68	I48	9.6I7	II	I.37
I6	400	332	336	296	I9.I67	2I	I.3I
24	600	498	504	444	28.455	30	I.25
34	850	692	706	622	45.277	42	I.23
48	I200	979	988	855	56.480	59	I.22

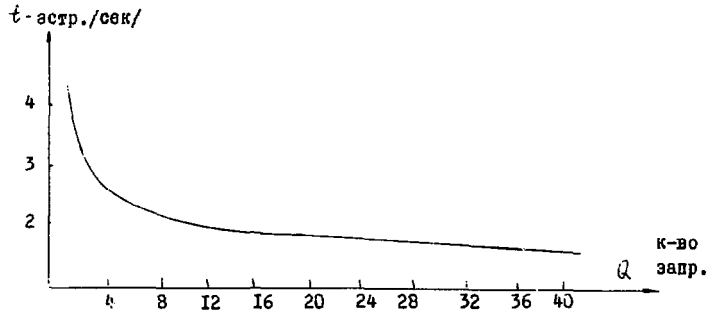

Рис.5. График зависимости относительного времени минимизации одного запроса от количества запросов

Л И Т Е Р А Т У Р А

I. Д.Лефковитц. Структура информационных массивов оперативных систем. М., Советское радио, 1973.

2. Т.Салтон. Автоматическая обработка, хранение и поиск информации. М., Советское радио , 1973.

3. ИПС "STAIRS", IBM, 1971, USA.

4. А.М.Китов. Программирование экономических и управленческих задач. М., Советское радио , 1971.

5. Д.А.Поспелов. Арифметические и логические основы цифровых машин. М., Энергия ,1964.

6. Д.Д.Арнаудов. Стратегия поиска в ИПС ОИЯИ. Сообщение ОИЯИ, Р10-8622, Дубна, 1975.

Рукопись поступила в издательский отдел
17 апреля 1975 г.