ТАРТУСКИЙ
ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

# ТРУДЫ

# ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА

47

ТАРТУ
1981

ТАРТУСКИЙ ГОСУДАРСТВЕННЫЙ
УНИВЕРСИТЕТ

# ОБРАБОТКА ДАННЫХ НА ЕС ЭВМ

## ТРУДЫ ВЫЧИСЛИТЕЛЬНОГО ЦЕНТРА

Выпуск 47

ТАРТУ 1981

Утверждено на заседании совета математического факультета ТГУ 23 октября 1981 года.

# ПРОГРАММНЫЕ СРЕДСТВА ОБСЛУЖИВАНИЯ ОБЩИХ БИБЛИОТЕК ПРОГРАММ

Ю. Кяхрик, Р. Роомельди, Т. Ээнма

Набором данных с библиотечной организацией в операционной системе ОС называется такой набор данных на внешних устройствах прямого доступа, который состоит из отдельных разделов и справочника, содержащего имена и адреса этих разделов. С точки зрения программного обеспечения особый интерес представляют библиотеки программ. Ясно, что от организации планирования, использования и обслуживания таких библиотечных наборов данных во многом зависит как производительность всей системы так и эффективность программирования.

В операционной системе ОС принято говорить о двух типах библиотек программ – системные библиотеки и библиотеки пользователей ЭВМ. Системными считаются такие библиотеки, которые создаются и заполняются в процессе генерации операционной системы. В качестве примеров можно назвать общую библиотеку программ SYS1.LINKLIB, макробиблиотеку SYS1.MACLIB, библиотеки подпрограмм трансляторов SYS1.FORTLIB, SYS1.PL1LIB и т.д. Во время эксплуатации ЭВМ такие библиотеки являются относительно постоянными, так как доступ пользователей в целях обновления разделов ограничен. Для восстановления сис-

темных библиотек достаточно иметь запасные копии и применять системные программы-утилиты.

Библиотеки пользователей создаются в процессе эксплуатации вычислительной системы при возникновении необходимости. В методике использования таких библиотек существует несколько разных подходов. Крайними из них являются следующие.

1. Нецентрализованное, частное владение библиотеками. В этом случае библиотеки называются личными и каждый пользователь (или группа пользователей) сам создает нужные ему библиотеки (подобно другим наборам данных) и обслуживает их. Такая методика обычно применяется при сеансовом стиле работы как в пакетном, так и в режиме с разделением времени. Своими библиотеками программисты-пользователи пользуются в монопольном режиме, защищая таким образом свои данные от других пользователей. Недостатками такого подхода являются: излишняя занятость программистов организационными работами при обслуживании библиотек, затрата внешней памяти на устройствах прямого доступа, частые прерывания в работе вычислительной системы.

2. Централизованное, общее владение библиотеками. В этом случае библиотеки пользователей создаются диспетчером системы в некотором оптимальном виде по отношению к вычислительному процессу в целом. Все библиотеки каталогизируются и программист должен знать лишь имена нужных ему библиотек. Сервисные работы над библиотеками выполняются обслуживающим персоналом ЭВМ. Преимущества такого подхода проявляется в эффективном использовании внешней памяти, в удобстве программирования и в относительной непрерывности вычислитель-

ного процесса.

Однако, в связи с общим использованием библиотек большим числом пользователей возникнут и некоторые сложные проблемы. Во-первых, операционная система не предоставляет возможностей защиты данных при параллельной обработке, в то время как разделы библиотек пользователей подвергаются частым обновлениям. Во вторых, такие библиотеки слишком велики, что влияет на время поиска разделов и увеличивает вероятность возникновения ошибок при обработке библиотек. И, наконец, аварии вычислительной машины или носителей внешней памяти могут разрушить значительную часть пользовательских библиотек, которые трудно восстановить.

Учитывая хотя бы указанные обстоятельства, ясно, что применение централизованного подхода требует более надежного программного обеспечения обслуживания библиотек чем то, которое содержит ОС в виде программ-утилит.

В вычислительном центре ТГУ имеются две ЭВМ ЕС-1022 одинаковой конфигурации, которые через устройство управления ЕС-5568 соединены в систему. Эта система обладает общей внешней памятью на трех накопителях ЕС-5061, компонентные ЭВМ имеют еще по три накопителя ЕС-5052 каждая. Общая внешняя память отведена в основном под системные и пользовательские библиотеки, что позволяет совместный доступ к этой информации с любой ЭВМ и дает возможность организовать нормальную работу при малом количестве внешних устройств прямого доступа. Библиотеки пользователей общие и обслуживаются централизованно. Введен ряд согласований по написанию имен программ, имен библиотек и т.д.

В дальнейшем рассматриваются некоторые программные средства, созданные в ВЦ ТГУ взамен или в дополнение к системным утилитам операционной системы ОС для обслуживания библиотек. Необходимость таких средств подтверждено практическим применением подхода общего использования библиотек. Более конкретно описываются программы обслуживания текстовых библиотек, копирования и сжатия любых библиотек и программы обслуживания архива (кроме того существует еще ряд вспомогательных программ, которые довольно специфические и не представляют общего интереса). Названные три программы нашли применение и в некоторых других вычислительных центрах Эстонии.

## 1. Обслуживание текстовых библиотек

Текстовыми называем в дальнейшем такие библиотеки, разделы которых состоят из записей фиксированной длины 80 байтов каждая и содержат информацию в символьном представлении. Примером можно назвать библиотеки исходных модулей языков программирования, макроопределений Ассемблера и процедур на языке управления заданиями. Относительный адрес начала раздела в виде TTR (тут и далее используются сокращения, общепринятые в ОС) запоминается в соответствующем элементе справочника, конец раздела определяется специальной записью EOD, которая следует последней записи в разделе.

Для обслуживания текстовых библиотек в операционной системе имеется программа-утилита IEBUPDTE, которая выполняет функцию корректировки - добавление новых разделов в библиотеку и обновление имеющихся разделов. Корректировка прово-

дится на основе управляющих предложений и обновленный раздел переписывается в конец библиотеки.

Рассмотрим некоторые вопросы надежности этой утилиты. Для лучшего понимания последующего приведем на рис. 1 принципиальную блок-схему работы IEBUPDTE, ограничиваясь для простоты случаем внесения изменений в имеющийся раздел.

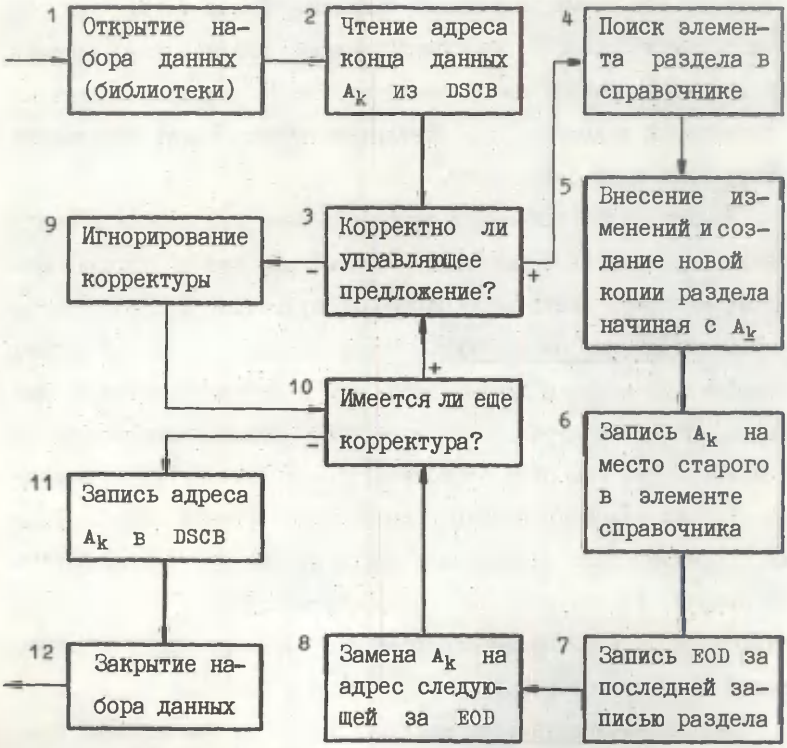



Рис. 1. Принципиальная блок-схема работы утилиты IEBUPDTE.

Как видно из схемы, в утилите отсутствуют средства защиты данных при одновременном использовании ее несколькими пользователями для изменения текстов в одной и той же библиотеке. Предполагается, что каждый пользователь обладает библиотекой монопольно. Это обстоятельство, однако, значительно ограничивает доступ к библиотекам, так как обладаемая монопольно библиотека остается недоступной другим пользователям на все время выполнения задания. Более того, при использовании библиотек из общей внешней памяти двумя системами (в нашем конкретном случае) вообще не существует никаких ограничений к доступу, в результате чего может произойти разрушение всей библиотеки.

Утилита IEBUPDTE имеет также серезные недостатки в своих реакциях на сбойные ситуации. Рассмотрим тут те случаи, которые наиболее часто встречаются на практике.

Переполнение библиотеки. Эта ситуация, когда при записи новой копии раздела оказывается, что в библиотеке нет больше места. В таком случае, как известно, задание заканчивается аварийно. Так как блок 6 рисунка 1 не выполняется, то в случае только одной корректуры сохраняется старая копия. Если же корректур было несколько и часть из них успели выполнить до аварии, то эти разделы в дальнейшем перекрываются, потому что блоки 11 и 12 не выполняются и в DSCB остается тот адрес конца данных, который там был до начала работы.

Авария вычислительной машины. Ситуация аналогична предыдущей, но может произойти и в случае только одной корректуры, когда работа прекращается после выполнения блока 6.

Сбой в критической точке. Как показывает практика, при

обработке длинных справочников вероятность возникновения сбоя носителя библиотеки наибольшая в момент между выполнением блоков 6 и 7 рисунка 1. В результате этого запись EOD не выполняется, а при дальнейшем использовании обновленного раздела оказывается, что к нему прицеплен следующий раздел и текст тем самым испорчен.

Учитывая указанные недостатки утилиты IEBUPDTE и повышенные требования к защите данных, пришлось разработать собственную программу для корректировки библиотек. Эта программа называется XUPDATE и она включена в состав программного обеспечения взамен утилиты IEBUPDTE. Далее описываются ее важнейшие отличия от стандартной утилиты и дается краткое содержание основных сообщений.

1.1. Управление ресурсами

Под ресурсами в данном контексте понимаются текстовые библиотеки, которые располагаются в общей внешней памяти и доступны разным операционным системам. Так как системы независимы, то доступ к библиотекам нельзя ограничивать системными средствами. Поэтому, при использовании программой XUPDATE все пользователи должны описывать соответствующие библиотеки разделяемыми (параметром DISP = SHR на языке управления заданиями). В системных процедурах это требование введено. Защиту данных гарантирует сама программа XUPDATE путем динамического захвата и освобождения ресурсов. При захвате ресурса (в результате чего разрешается доступ к библиотеке) используются возможности системной макрокоманды RESERVE. Для освобождения ресурса выполняется макрокоманда DEQ. На рис. 2 приве-

ден отрезок программы управления ресурсом.

```
* НАХОЖДЕНИЕ АДРЕСА
         SR       2,2
         EXTRACT TIOTADDR,FIELDS=TIOT
         L        3,TIOTADDR
         LA       3,24(3)
NXTDD    CLC      4(8,3),=CL8'SYSUT2'
         BE       FINDUCB
         IC       2,Ø(3)
         AR       3,2
         CLC      Ø(4,3),=F'Ø'
         BNE      NXTDD
         ABEND    18
FINDUCB  LA       4,16(3)
* ЗАХВАТ РЕСУРСА
         RESERVE (QNAME,RNAME,E,44,SYSTEMS),UCB=(4)
         ...
* ОСВОБОЖДЕНИЕ РЕСУРСА
         DEQ      (QNAME,RNAME,44,SYSTEMS)
         ...
* КОНСТАНТЫ
TIOTADDR DS       F
QNAME    DC       C'DATASETS'
RNAME    DS       CL44    ИМЯ БИБЛИОТЕКИ
```

Рис. 2. Программа управления ресурсом.

Именем ресурса служит имя библиотеки. Следует подчеркнуть, что если в пределах одной операционной системы во время обладания ресурсом конкурирующим пользователям запрещаются лишь операции записи в эту библиотеку, то в отношении другой системы блокируется весь носитель данных. Это обстоятельство требует сокращения до минимума времени обладания ресурсом. В программе XUPDATE, например, такие функции как

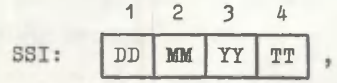
проверка правильности управляющих предложений, выполняются до открытия библиотеки.

В практике оказалось полезным извещать оператора ЭВМ о ходе обладания ресурсами. Соответствующие сообщения приведены ниже.

### 1.2. Надежность программы

При централизованном обслуживании библиотеками программисты-пользователи освобождены от всех сервисных работ. Их интересует лишь получение из библиотеки верного текста, при условии, что они сами не нарушают установленных соглашений. Под надежностью обслуживающей программы и понимается ее способность гарантировать верность информации как при нормальной работе, так и в разных сбойных ситуациях.

В целях повышения надежности программы XUPDATE в ней происходит снабжение текстов библиотеки признаком версии. Для этого используется 4-байтовое поле в элементе справочника (известное как поле SSI), которое заполняется следующей информацией




где DD - день, MM - месяц, YY - последние две цифры года и TT - время (с точностью до полного часа) обновления текста или добавления нового текста в библиотеку. На основе этой информации программистам и обслуживающему персоналу легко следить за правильностью версий.

Рассмотрим еще работу программы XUPDATE в тех неприятных случаях, которые были описаны выше при указании недостатков

утилиты IEBUPDTE. Чтобы предотвратить ситуацию, возникающую при переполнении библиотеки, весь поток корректур искусственно разбивается на части. Каждая часть содержит изменения только одного раздела. Таким образом, блок-схема рисунка 1 (кроме блоков 3 и 9) применяется поочередно несколько раз. В результате несколько увеличивается время обработки, но надежность повышается этим простым способом значительно.

Проблему корректной фиксации конца раздела решить сложнее. Так как блоки 6 и 7 (на рис. 1) в действительности выполняются в среде макрокоманды STOW, то трудно следить за выполнением этих шагов в отдельности. В программе XUPDATE используется способ повторной обработки конца текста. Во все справочники текстовых библиотек введен некоторый фиктивный элемент и способ повторной обработки заключается в том, что обновленный раздел читается фиктивным, а после успешного завершения макрокоманды STOW ее же выполняют снова с действительным именем. Иллюстрируем сказанное блок-схемой на рисунке 3, где номера основных блоков взяты с рисунка 1.

В этой схеме требуют объяснения выходы A и B. Если реализуется выход A, то корректура просто игнорируется и в библиотеке остается старая копия текста. Если же реализуется выход B, то в библиотеке остается корректная копия обновленного раздела, так как блок EOD записан заранее. Следует отметить, что в нормальном случае раздел заканчивается также одним блоком EOD. Имя фиктивного элемента следует выбирать так, чтобы соответствующий элемент в справочнике стоял в качестве первого (в конкретном случае этим именем является @@@@@@@@): тогда требуется минимальное время поиска. Исполь-

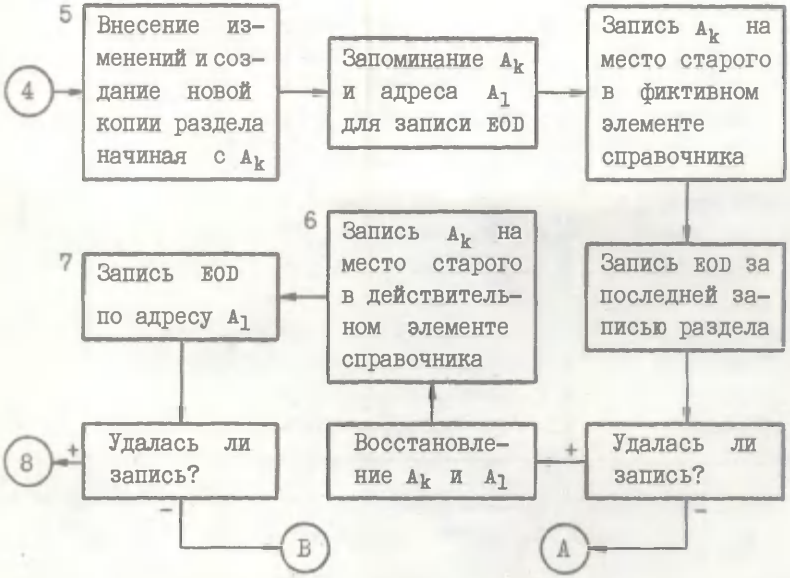



Рис. 3. Расширение блок-схемы повторной обработки.

зовать это фиктивное имя в других целях запрещено.

1. 3. Сокращенные управляющие предложения

Управляющие предложения утилиты IEBUPDTE содержат ряд ключевых параметров, написание и перфорирование которых довольно трудоемко. Поэтому для программы XUPDATE введена более удобная форма написания параметров, хотя при желании программист может использовать и стандартную форму. На рисунке 4 представлены короткие и соответствующие им стандартные формы написания управляющих предложений программы XUPDATE (стандартные формы полностью соответствуют предложениям утилиты IEBUPDTE). Следует отметить, что в короткой форме записи лишние пробелы не допускаются.

| Короткая форма | Стандартная форма |
|---|---|
| ./␣A␣<ИМЯ>[,L] | ./␣ADD␣NAME=<ИМЯ>[,LIST=ALL] |
| ./␣C␣<ИМЯ>[,{L / I}] | ./␣CHANGE␣NAME=<ИМЯ>[,{LIST=ALL / UPDATE=INPLACE}] |
| ./␣R␣<ИМЯ>[,L] | ./␣REPRO␣NAME=<ИМЯ>[,LIST=ALL] |
| ./␣RL␣<ИМЯ>[,L] | ./␣REPL␣NAME=<ИМЯ>[,LIST=ALL] |
| ./␣AL␣<ИМЯ> | ./␣ALIAS␣NAME=<ИМЯ> |
| ./␣N␣n | ./␣NUMBER␣NEW1=n,INCR=n |
| ./␣N␣ALL,n | ./␣NUMBER␣SEQ1=ALL,NEW1=n,INCR=n |
| ./␣N␣k,1,m | ./␣NUMBER␣SEQ1=k,NEW1=I,INCR=m,INSERT=YES |
| ./␣D␣k,I | ./␣DELETE␣SEQ1=k,SEQ2=1 |
| ./␣D␣k | ./␣DELETE␣SEQ1=k,SEQ2=k |

Рис. 4. Таблица соответствия коротких и стандартных форм записи управляющих предложений корректирования.

Как отмечалось выше, управляющие предложения рассматриваются программой XUPDATE до начала процесса корректирования. При обнаружении ошибок выводится соответствующее предупредительное сообщение на пульт оператора и на печатающее устройство. Действие, требуемое некорректным предложением игнорируется и корректура пропускается. Такие ошибки появляются обычно в результате неправильной перфорации и они устраняются обслуживающим персоналом, не сообщая об этом программисту.

Как известно, для замены и вставления строк в текст утилита IEBUPDTE требует указания 8-значного порядкового номера строки в конце строки (позиции 73-80). Из-за неудобств перфорирования таких строк для программы XUPDATE достаточно добавить лишь один пробел к информации в строке, символ-разде-

литель 'Й' и за этим номер строки. При этом начальные нули можно пропустить. Признаком продолжения строки в Ассемблере служит символ '*', который пишется непосредственно перед номером строки. Например, перфорированные строки

```
        LA␣␣␣3,4␣Й5Ø
DCB␣␣␣␣␣␣DCB␣␣DSORG=PO,␣Й*12Ø
```

идентичны строкам

```
        LA␣␣␣3,4␣                                   ØØØØØ5Ø
DCB␣␣␣␣␣␣DCB␣␣DSORG=PO,                            *ØØØØ12Ø
```

1.4. Сообщения программы

Сообщения программы XUPDATE информируют оператора ЭВМ о ходе работы программы и пользователей об ошибках, или же представляют собой просто справочную информацию. Сообщения в основном понятны без объяснений. Ниже приводятся некоторые характерные сообщения.

<имя библиотеки>␣RES(<имя задания>,<имя текста>)

Это сообщение выдается на пульт оператора и информирует о захвате ресурса. Если имя библиотеки имеет вид SYS1.xxxLLB, то в сообщении печатается только xxx (так как в нашем вычислительном центре все имена библиотек подобного вида стандартизованы).

<имя библиотеки>␣FREE,TTR=tttrr,FREETTR=t't't'r'r'

Это сообщение выдается на пульт оператора при освобождении ресурса. Параметр TTR указывает относительный адрес обновленного текста, FREETTR указывает адрес свободного участка в библиотеке. В режиме UPDATE=INPLACE эти параметры от-

сутствуют, они также отсутствуют в том случае, когда корректура была неуспешной (тогда указывается код завершения). Следует отметить, что эти два сообщения позволяют следить за процессом обновления библиотеки и за состоянием обрабатываемых разделов.

BEFORE CORRECTIONS IN LIBRARY <имя библиотеки>
<имя текста> <дата> <время> <управл. предложение> <оценка>

Такая таблица выдается на печать и информирует пользователя о результатах просмотра управляющих предложений. Например, если таблица имеет вид

```
XYKGR     21.01.81  15.00 CHANGE TRUE
XYKIN     NO DATE         ADD    FALSE PARM IS NOT NEW
XYKSTØ1   NOT IN LIBRARY  REPL   FALSE CHANGED TO ADD
XYKTP     NOT IN LIBRARY  CHANGE TRUE  TRANSMITTING FROM
                                       ARCHIVES NEEDED
```

то текст XYKGR содержится в библиотеке (последнее обновление было 21 января 1981 в 15 часов). Новое обновление (CHANGE) возможно, что и подтверждает слово TRUE. Текст XYKIN также содержится в библиотеке, но в каталоге отсутствует дата (NO DATE). Поскольку не включен режим PARM=NEW, то добавление (ADD) нового текста с таким же именем невозможно (FALSE). Текст XYKSTØ1 не содержится в библиотеке. Поскольку операцией является замена (REPL), то не имеет смысла восстановить этот текст из архива. Поэтому REPL заменяется на ADD. Текст XYKTP также не содержится в библиотеке, но операцией является обновление (CHANGE). Поэтому добавляется сообщение о необходимости восстановления этого текста из архива.

XUP (<имя задания>) SYSIN=(n;m,k) <текст сообщения>

Такое сообщение, выдаваемое на пульт оператора и на печать, информирует об обнаружении ошибки в потоке корректуры. Для облегчения поиска ошибочной карты в колоде перфокарт задания указывается точное место: n = порядковый номер карты в SYSIN, m = порядковый номер корректуры, k = порядковый номер карты в корректуре. Во второй строке печатается содержимое ошибочной карты. Текст сообщения может быть одним из следующих:

FIRST CARD IS NOT CONTROL COMMAND
'NAME=' MISSING IN CONTROL COMMAND
INVALID CHARACTER IN NAME FIELD
<имя текста> CORRECTION IGNORED, NUMBER IN
SECOND CARD MISSING
NUMBER LONGER THAN 8 CHARACTERS
NUMBER MISSING
NUMBER IS NOT DECIMAL

Вопросы, касающие автоматического использования архива в ходе работы программы XUPDATE будут рассмотрены ниже.

## 2. Копирование и сжатие библиотек

Для копирования библиотечных наборов данных или отдельных разделов обычно используется системная утилита IEBCOPY. Сжатием называется такой вид копирования, где входная и выходная библиотека одна и та же. Сжатие выполняется с целью освобождения неиспользованной памяти в библиотеке, так как в процессе сжатия старые копии разделов удаляются. Если биб-

лиотеки используются и обновляются большим числом программистов, то приходится их довольно часто сжимать. Этим и объясняется большой интерес к работе утилиты IEBCOPY.

При выполнении сжатия библиотеки рекомендуется сохранить ее копию до тех пор, пока не будет выдано сообщение о конце задания с кодом возврата ØØ. Это связано с тем, что справочник библиотеки для повышения скорости обрабатывается вс внутренней памяти, а разделы копируются непосредственно в библиотеку. Таким образом, логические связи между элементами справочника и разделами в библиотеке временно нарушены и в случае аварии в это время библиотека оказывается непригодной для дальнейшей работы (это видно из принципиальной блок-схемы работы утилиты, приведенной на рис. 5). Такая ситуация может возникнуть и при обычном копировании, но тогда приходится просто выполнить задание заново.

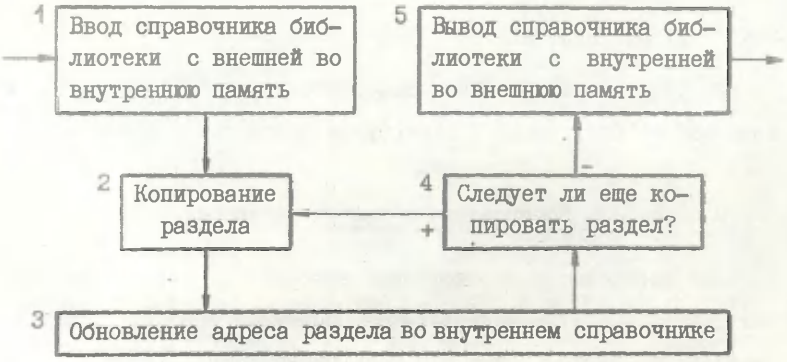



Рис. 5. Принципиальная блок-схема копирования библиотек утилитой IEBCOPY.

При малом объеме внешней памяти на дисковых устройствах приходится разгрузить библиотеку перед каждым сжатием на магнитную ленту и в случае аварии восстановить. Чтобы избавиться от этой трудоемкой работы, в ВЦ ТГУ и была разработана собственная программа копирования.

Следует отметить еще некоторые менее важные, но все-таки мешающие нормальной работе недостатки утилиты IEBCOPY. Во первых, отсутствует обработка ошибок (сбоев) внешнего устройства - накопителя библиотеки. При появлении сбоя утилита просто заканчивает работу с некоторым кодом возврата, который указывает лишь причину ошибки. Для продолжения приходится повторить всю работу, пропустив тот раздел, при обработке которого возник сбой. Во вторых, выходной листинг утилиты объемист, но не содержит всю нужную информацию. К тому же отсутствует возможность управления печатью разных списков и служебной информацией.

Для копирования и сжатия библиотек в ВЦ ТГУ разработана и внедрена новая программа PW2COPY. Принципиальным отличием этой программы от утилиты IEBCOPY является обработка элементов справочника прямо в библиотеке. Скорость копирования от этого несколько падает, но увеличивается надежность работы - не требуется страхование библиотек в запасной памяти, так как в случае аварии потеряется максимально только один раздел. При сжатии библиотек для обновления справочника используется метод EXCP, в связи с чем программа PW2COPY в процессе сжатия не уступает утилите в скорости.

Возможности программы PW2COPY расширены, учитывая конкретные потребности пользователей и особенности работы с

большими библиотеками. Например, можно управлять действиями программы в случае сбоев, обнаруживать и сообщать оператору ЭВМ нарушения логической структуры библиотеки и восстановить ее, переблокировать записей и т.д.

Так как использование программы PW2COPY и утилиты IEBCOPY в общем сходятся, то ниже мы ограничиваемся описанием тех различий, которые свойственны программе PW2COPY.

2.1. Управление программой

Описание входного и выходного библиотечного набора данных на языке управления заданиями, а также операторы DD с именами SYSPRINT и SYSIN для программы PW2COPY и утилиты IEBCOPY одинаковы. Вспомогательных наборов данных (описываемых для утилиты операторами DD с именами SYSUT3 и SYSUT4) программа PW2COPY не требует.

Для управления печатью введен необязательный оператор DD с именем PRINT. Если он присутствует в потоке задания, то справочники копируемых библиотек выводятся на печать постранично, соблюдая 72-строковый формат. Сообщения программы выводятся на SYSPRINT.

Для управления работой программы PW2COPY используются управляющие предложения COPY, SELECT, REPLACE, EXCLUDE, LISTPDS, CONSOLE, GO, END и CC. Предложения можно вводить с перфокарт в потоке заданий, с пульта оператора или с поля параметров (в последнем случае только одно предложение). На перфокарты они перфорируются на позициях 1-80, продолжение запрещено. При желании управляющие предложения и коды операций можно сократить до одной (первой) буквы (кроме *CONSOLE*,

GO, END, CC, YES, NO), например, управляющее предложение COPY можно представить в виде

```
C␣I=DD1,O=DD2,E=E,A=NO,S=YES
```

Управляющее предложение COPY требуется для копирования библиотек и ее полный формат таков:

```
COPY␣INDD=<имяdd>,OUTDD=<имяdd>[,REPL={YES|NO}]
[,EROPT={CANCEL|EXCLUDE|IGNORE|WTOR}][,MAXE=nn]
[,LIST=l1l2l3][,ATLAS={YES|NO}][,SECALLOC={YES|NO}]
```

Параметры имеют следующие значения:

INDD - определяет имя оператора DD для входной библиотеки;

OUTDD - определяет имя оператора DD для выходной библиотеки; для сжатия библиотеки это имя должно совпадать с именем в параметре INDD; разными именами в параметрах INDD и OUTDD нельзя описывать одну библиотеку;

REPL - определяет режим замены разделов, причем YES указывает, что все последующие предложения SELECT рассматриваются как REPLACE или же при полном копировании разрешается замена разделов в выходной библиотеке;

EROPT - указывает действия в сбойных ситуациях: CANCEL означает, что работа прекращается; при значении IGNORE сбой игнорируется и в зависимости от характера сбоя или пропускается блок (дорожка), или раздел копируется неполностью, или обрабатывается неполный справочник; в случае значения

EXCLUDE раздел не копируется (при сжатии удаляется с библиотеки); значение WTOR означает, что оператор ЭВМ выбирает один из описанных вариантов;

MAXE - устанавливает наибольшее число nn некопируемых разделов в случае сбоя в режиме EROPT=EXCLUDE;

LIST - указывает режим печати справочников входной библиотеки ($1_1$), выходной библиотеки перед копированием ($1_2$) и после копирования ($1_3$) следующим образом: 0 = не печатать, 1 = печатать в алфавитном порядке, 2 = печатать в физическом порядке, 3 = печатать в режимах 1 и 2 (например, LIST = 302 требует печати справочника входной библиотеки в алфавитном и физическом порядке, справочника выходной библиотеки после копирования в физическом порядке);

ATLAS - указывает, разрешено ли (YES) назначение альтернативной дорожки до действия по операнду EROPT;

SECALLOC - определяет действие программы при переполнении выходной библиотеки: YES указывает, что библиотеку можно расширить на дополнительные экстенты и продолжить копирование; по значению NO копирование прекращается (это значение присваивается автоматически, если библиотека создана параметром CONTIG или все 16 экстентов уже использованы).

Для выборочного копирования разделов используются управляющие предложения SELECT, REPLACE и EXCLUDE. В предложениях SELECT и REPLACE указываются имена или промежутки разделов во входной библиотеке, которые принадлежат копированию. При этом REPLACE разрешает заменить копируемые разделы в выходной библиотеке, а SELECT нет (если не указан режим REPL=YES). Предложением EXCLUDE можно указать имена или промежутки тех

разделов, которые не копируются. Это предложение можно использовать и при сжатии для стирания разделов с библиотеки.

Формат управляющих предложений выборки таков:

$$\left\{\begin{matrix}\text{SELECT}\\ \text{REPLACE}\\ \text{EXCLUDE}\end{matrix}\right\}\sqcup\left\{\begin{matrix}\text{MEMBER=<имя}_1\text{>[,<имя}_2\text{>,...]}\\ \text{MEMBER=(<имя}_v\text{>=<имя}_u\text{>)[,...]}\\ \text{AS=(<имя}_a\text{>-<имя}_1\text{>)[,...]}\\ \text{PHS=(<имя}_a\text{>-<имя}_1\text{>)[,...]}\end{matrix}\right\}$$

где MEMBER определяет выборку по именам:

$имя_1$, $имя_2$ задают выбираемые имена разделов,

$имя_v$ – имя копируемого раздела при SELECT или REPLACE, которые надо заменить ("старое" имя),

$имя_u$ – имя, присваиваемое копированному разделу в выходной библиотеке ("новое" имя);

AS и PHS определяют промежуточную выборку с $имя_a$ до $имя_1$ (AS указывает алфавитный порядок имен разделов, PHS физический порядок).

Управляющее предложение LISTPDS, используемое для печати и проверки корректности справочников библиотек, имеет формат:

$$\text{LISTPDS}\sqcup\text{INDD=<имяdd>}[\text{,LIST=l}_1][\text{,ATLAS=}\left\{\begin{matrix}\text{YES}\\ \text{NO}\end{matrix}\right\}]$$

$$[\text{,EROPT=}\left\{\begin{matrix}\text{CANCEL}\\ \text{IGNORE}\\ \text{WTOR}\end{matrix}\right\}]$$

где назначение параметров то же, что в случае предложения COPY. Добавим лишь, что во входном потоке можно указать любое количество управляющих предложений COPY и LISTPDS.

Иногда удобно управлять процессом копирования или сжатия с пульта оператора. Для переключения источника управляющих

предложений на пульт оператора служит управляющее предложение CONSOLE (без параметров). Программа реагирует на это предложение выдачей сообщения

'ENTER CONTROL STATEMENT' ,

после чего с пульта оператора можно вводить любые управляющие предложения. Запустить их можно управляющим предложением GO или END (в случае GO после выполнения программы PW2COPY можно набрать следующий набор управляющих предложений, в случае END работа программы заканчивается). Аннулировать введенные управляющие предложения можно предложением CC.

В таблице на рис. 6 приведены значения параметров управляющих предложений COPY и LISTPDS, принимаемые по умолчанию.

| | COPY (между разными библиотеками) | COPY (сжатие) | LISTPDS |
|---|---|---|---|
| INDD | требуется | требуется | требуется |
| OUTDD | требуется, отличный от INDD | требуется, должен совпадать с INDD | игнорируется |
| REPL | NO | игнорируется | игнорируется |
| EROPT | EXCLUDE | WTOR | IGNORE |
| | WTOR, когда управляющие предложения вводятся с пульта оператора | | |
| MAXE | 5 | 3 | игнорируется |
| | при EROPT=WTOR принимается MAXE=32760 | | |
| LIST | 001 | 302 | 3 |
| ATLAS | YES | YES | NO |
| SECALLOC | NO | игнорируется | игнорируется |

Рис. 6. Таблица значений параметров программы PW2COPY.

2.2. Сообщения программы PW2COPY

Сообщения выдаются на печать, а некоторые из них дублируются на пульт оператора. Каждое сообщение начинается фразой CPYxxx, где xxx номер сообщения. В программе PW2COPY имеется 113 разных сообщений, из которых здесь описываем лишь те, которые наиболее часто встречаются в практике и иллюстрируют работу программы.

CPYØ18 SEVERITY CODE WAS xx

Такое сообщение выдается в конце выполнения некоторой функции программы и указывает код возврата: ØØ = успешное завершение; Ø2 = ошибка в управляющем предложении; Ø4 = сбой, некоторый раздел копирован частично (EROPT=IGNORE); Ø8 = сбой, некоторый раздел не копирован (EROPT=EXCLUDE); 12 = мало памяти, приходится увеличивать значение параметра REGION; 16 = функция прекращена из-за сбоев.

CPYØ21 THE {IN / OUT} PUT PDS:

В этом сообщении после копирования или сжатия указываются параметры библиотеки в следующем виде:

DDNAME=<имяdd> VOL=SER=<имя тома> DSNAME=<имя библиотеки>

NO. OF EXTENTS <число экстентов>

NO. OF ALLOCATED TRACKS <число дорожек в библиотеке>

NO. OF UNUSED TRACKS <число свободных дорожек>

CPYØ52 TTR IN DIRECTORY ERRONOUS FOR MEMBER (<имя раздела>)

Адрес раздела во входным справочнике неправилен, причиной чему может быть, например, то, что первому блоку раздела не предшествует EOD, или адрес раздела меньше чем адрес EOD

предыдущего раздела, или адрес раздела выходит за пределы библиотеки. Последнюю ситуацию невозможно игнорировать параметром EROPT=IGNORE.

CPYØ87 MINIMUM I/O BUFFER (xxxxxK) NOT ALLOCATABLE

Мало буферной памяти - значение параметра REGION следует увеличить как минимум на xxxxxK байтов.

CPYØ92 I/O ERROR READING MEMBER (xxxxxxxx)-<характер ошибки>

Сбой при считывании данных раздела xxxxxxxx. Реакция на сбой стандартна. При EROPT=IGNORE пропускается один блок или дорожка.

CPYØ92 I/O ERROR WRITING MEMBER (xxxxxxxx)-<характер ошибки>

Сбой при записи данных раздела xxxxxxxx. Операцию пытаются повторить 5 раз, после этого работа прекращается.

CPY1Ø2 REPLY YES OR NO TO EXCLUDE n MEMBERS
FROM <имя библиотеки>

Сообщение выдается только на пульт оператора. Спрашивается разрешение к стиранию n разделов в указанной библиотеке при сжатии. По ответу NO сжатие прекращается. Список имен этих разделов указывается в сообщении CPYØ54.

CPY1Ø3 MAIN STORAGE LACK - INCREASE REGION

Мало памяти для ввода справочников.

CPY1Ø7 REPLY YES OR NO TO CORRECT THE SEQUENCE ERRORS

По ответу YES корригируется справочник, притом от тождественных имен удаляют те, относительный адрес (TTR) которых меньше.

CPY111 MEMBER xxxxxxxx TTR=ttttrr - REFERENCE
INTO MEMBER yyyyyyyy

Это сообщение выдается только при EROPT=WTOR (в остальных случаях CPYØ52) и указывает, что адрес раздела xxxxxxxx ссылает в раздел yyyyyyyy. Следует сообщение CPY112.

CPY112 REPLY YES OR NO TO SEPARATE MEMBERS

Ошибка, описанная предыдущим сообщением возникла вероятно от того, что отсутствует EOD раздела yyyyyyyy. По ответу YES записывается EOD по адресу ttttrr-1. По ответу NO переписывается раздел yyyyyyyy.

## 3. Обслуживание архива

В ходе эксплуатации ЭВМ происходит увеличение числа программ в библиотеках пользователей. Следовательно, время от времени приходится увеличивать размеры библиотек. Когда же во внешней памяти больше нет места, следует просмотреть содержание библиотек. Оказывается, что число программ, с которыми программисты активно занимаются (отлаживают и изменяют), увеличилось, как правило, незначительно, а библиотеки в основном содержат пассивные программы, которые в данное время их авторов не интересуют. Ясно, что таких программ нет смысла держать в библиотеке. Стирать, однако, можно только некоторые из них (например, студенческие программы после окончания семестра), так как программисты обычно хотят сохранить по крайней мере исходные тексты: в отлаженных программах могут обнаружиться ошибки и приходится изменять тексты программ. Следовательно, вопрос заключается во временном удале-

нии программ из библиотек и в восстановлении их при надобности.

В ВЦ ТГУ внешних устройств прямого доступа сравнительно мало, поэтому считается целесообразным располагать такие временно ненужные программы на магнитных лентах, создавая таким образом архив.

Структура архива и соответствующие программы обслуживания описываются ниже, сперва остановимся коротко на способах удаления программ из библиотек.

Библиотеку можно считать нормально нагруженной, когда ее свободный участок после сжатия составляет примерно одну четверть от всей библиотеки. Для проверки длины активных библиотек создана специальная процедура, которую оператор ЭВМ в ходе работы периодически применяет. Если окажется, что свободного места слишком мало, некоторые разделы приходится удалить из библиотеки. Способ удаления зависит от типа библиотеки.

В текстовых библиотеках оценкой активности раздела служит дата, помещенная в элемент справочника программой XUPDATE (см. п. 1.2). Получив список имен разделов в порядке возрастания дат (такой список выдает управляющее предложение LISTPDS программы PW2COPY), диспетчер ЭВМ определяет ту границу, начиная от которой разделы считаются пассивными (выбор границы зависит от многих факторов: от общей нагруженности библиотеки, длины текстов и т.д.). После установления границы вся библиотека выгружается на магнитную ленту и пассивные разделы удаляются из библиотеки либо специальной программой, либо в процессе сжатия программой PW2COPY.

Несколько иначе обрабатываются библиотеки модулей. Под модулями здесь понимаются такие программы пользователей, полученные в результате полной или неполной обработки Редактором связей, которые прямо не подлежат загрузке, а используются для объединения с другими аналогичными. Активность модуля зависит от двух компонент: от даты последнего обновления (редактирования) и от частоты использования модуля в качестве подпрограммы. В ВЦ ТГУ сделано дополнение к программе редактирования модулей IEWL, которое помимо управления ресурсом позволяет поместить в элемент справочника модуля дату редактирования. Содержимое этого 4-байтового поля совпадает с полем SSI в справочнике текстовой библиотеки, только байт TT является счетчиком частоты использования модуля и при редактировании модуля там записывается 0.

В практике оказывается полезным разделить библиотеки модулей на два класса: активные библиотеки модулей и библиотеки пользовательских подпрограмм. Часто редактируемые модули содержатся в активной библиотеке (соответствующие тексты, по-видимому, тоже подлежат частым обновлениям), а библиотеку подпрограмм использует Редактор связей в качестве входной библиотеки для поиска нужных подпрограмм. В ВЦ ТГУ создана одна активная библиотека модулей SYS1.MODUL.LINK и библиотека подпрограмм SYS1.MODUL.SUB . В процедурах неполного редактирования в качестве выходной библиотеки определена SYS1.MODUL.LINK:

```
//SYSLMOD DD DSN=SYS1.MODUL.LINK(<имя модуля>),DISP=SHR
```

а в процедурах полного редактирования в качестве вызываемой библиотеки SYSLIB задается следующая цепь:

```
//SYSLIB DD DSN=SYS1.MODUL.LINK,DISP=SHR
//        DD DSN=SYS1.MODUL.SUB,DISP=SHR
```

(к которым при надобности прицепляют еще библиотеки подпрограмм языков программирования).

Когда некоторый модуль из библиотеки подпрограмм используется при редактировании, то автоматически прибавляется единица к частоте ее использования (частота 255 не меняется).

По мере заполнения активной библиотеки часть модулей переписывается в библиотеку подпрограмм и библиотека очищается по датам последнего обновления как и текстовые библиотеки. Активная библиотека модулей в архив не выгружается: когда заполняется библиотека подпрограмм, то запускается специальная программа сравнения библиотек и удаляются из нее те модули, которые вновь появились в библиотеке активных модулей. Если после этого все-таки мало места в библиотеке подпрограмм, то упорядочивается список имен модулей по возрастанию частот, библиотека выгружается в архив и удаляются те модули, частота которых невелика (выбор удаляемых модулей проведет диспетчер). У оставшихся модулей байт частоты приравнивается нулю. В дальнейшем архивные файлы модулей налаживаются аналогично текстовым.

### 3.1. Структура архива

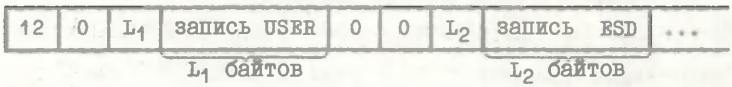
Архивом называется совокупность последовательных наборов данных на магнитных лентах, содержащих разделы из библиотек. Каждой библиотеке при этом соответствует один архивный файл, который в общем случае состоит из нескольких наборов данных на разных магнитных лентах. В пределах одного набора данных

разделы упорядочены в порядке убывания даты. В ходе работы может случиться, что некоторый раздел представлен в нескольких наборах данных одного архивного файла. Поэтому периодически выполняется специальная программа упорядочения архивного файла, которая переформирует наборы данных так,что первый набор содержит самые новые разделы, следующий более старые и т.д. Следует отметить, что в практике для восстановления разделов, как правило, требуется только первый набор данных архивного файла.

Каждый раздел начинается в архивном файле 80-байтным инфоблоком, за которым следуют записи раздела блоками по 8000 байтов каждая (последний блок раздела может быть короче). В инфоблоке указываются имя раздела, порядковый номер этого блока в наборе данных, дата раздела, тип раздела и содержимое поля TTRC, взятое из библиотеки. Кроме того все разделы одного набора данных связаны ссылками и в инфоблоке содержится имя предыдущего раздела вместе с некоторой контрольной информацией (число байтов и содержимое первых и последних восьми байтов предыдущего раздела). Конец набора данных обозначается аналогичным инфоблоком, только именем является там **ENDI** и контрольные поля содержат информацию о последнем разделе.

В архивном файле текстовой библиотеки данные объединены в блоки в виде 80-байтных логических записей. В случае модулей структура блоков несколько сложнее: так как логические записи модуля имеют переменную длину и относятся к некоторому типу, то эти данные необходимо зафиксировать, чтобы можно было восстановить модуль. Каждая логическая запись модуля в

архивном файле начинается 4-байтной меткой, где указывается тип записи (0 для записей ESD, 4 для TXT и 12 для поля USER справочника), признак NOTELST для записей типа TXT (0 = не является элементом списка NOTELST, 4 = соответствующий TXT является элементом списка NOTELST) и длина записи в байтах. Если длина записи равняется нулю, то это является признаком конца раздела. Логическая запись может продолжаться в следующем физическом блоке. Таким образом, первый блок модуля выглядит всегда следующим образом:




и последний блок кончается нулевой меткой.

3.2. Управление программой обслуживания архива

Функциями используемой в ВЦ ТГУ программы XYKCOPY являются выгружение библиотеки в архив, восстановление библиотеки и соединение наборов данных архивного файла. Программа может обработать также части библиотек или архивных наборов данных. Функция восстановления выполняется только в том случае, когда по какой-то причине разрушена вся библиотека или значительная часть ее, для восстановления отдельных разделов используется более специфичная программа XYKRES, которую будем рассматривать при описании автоматического восстановления.

Управление программой XYKCOPY осуществляется управляющими предложениями на языке управления заданиями, описывающими наборы данных и специальными управляющими предложениями программы, определяющими выполняемые функции. Для печати таблиц

используется дополнительный системный вывод PRINT, а управление печатью производится значениями $l_1 l_2 l_3$ также, как и в случае программы PW2COPY.

Управляющие предложения программы перфорируются на карты начиная с позиции 2. Продолжения и излишние пробелы не допускаются. При обнаружении ошибок работа программы заканчивается.

Библиотека выгружается в архив по управляющему предложению C, формат которого:

␣C␣I=<имядd>,O=<имядd>

где I определяет имя предложения DD выгружаемой библиотеки и O – имя DD набора данных архивного файла. Следует отметить, что этот набор данных всегда создается. Разделы библиотеки записываются на магнитную ленту в таком порядке, как они располагаются в библиотеке (обычно в порядке убывания даты). Структуру архива (текстовую или модульную) выбирает сама программа XYKCOPY по типу библиотеки.

Для восстановления библиотеки используется управляющее предложение

$$\text{␣R␣I=}\left\{\begin{array}{l}\text{<имядd>}\\ \text{(<имядd>,R)}\end{array}\right\}\text{,O=<имядd>}$$

Тут I определяет набор данных архивного файла, а O – библиотеку. Параметром R пользуются тогда, когда восстановление происходит в имеющуюся библиотеку и необходимо перекрыть одноименные разделы восстанавливаемыми.

Для объединения двух наборов данных архивного файла или для создания новой копии набора данных используется предложение

$$\text{␣T␣I=<имя dd>,O=}\left\{\begin{array}{l}\text{<имя dd>}\\ \text{(<имя dd>,R)}\end{array}\right\}$$

где I и O определяют имена предложений DD соответственно для входного и выходного набора данных архивного файла. Если выходным является новый набор данных, то помимо DISP=NEW необходимо указать и параметр R. При объединении сначала составляется список разделов выходного набора и прибавляются лишь те разделы из входного набора данных, которых раньше не было.

Для выборочного выполнения вышеуказанных функций применяются взаимно исключающие управляющие предложения S (включить) и E (исключить). Выборку разделов можно задавать списком имен в виде

$$\text{␣S␣M=<имя}_1\text{>}\,[\text{,<имя}_2\text{>,...}]$$

$$\text{␣E␣M=<имя}_1\text{>}\,[\text{,<имя}_2\text{>,...}]$$

или промежутком

$$\text{␣S␣N=(<имя}_a\text{>-<имя}_1\text{>)}$$

$$\text{␣E␣N=(<имя}_a\text{>-<имя}_1\text{>)}$$

Промежуток выбирается по порядку действительного расположения разделов в библиотеке или в наборе данных. Если опущено начальное или конечное имя промежутка (N=(-<имя$_1$>) или N=(<имя$_a$>-)), то обрабатываются все разделы соответственно до указанного имени или от указанного имени.

### 3.3. Автоматическое восстановление разделов

Важной проблемой при использовании архивом программ является восстановление нужных разделов в библиотеке без прекращения задания пользователя. Для этого следует сперва узнать, какие тексты или модули требуются в конкретном задании. В ВЦ

ТГУ реализован просмотр библиотеки в программах корректора XUPDATE и Редактора связей IEWL. В корректоре обрабатывается библиотека, задаваемая в качестве SYSUT2, а для Редактора связей требуется дополнительная информация - пользователь должен описать используемые модули в управляющих предложениях INCLUDE. Тогда организуется поиск по указанным библиотекам, увеличивается частота модулей в библиотеке подпрограмм и строится список ненайденных модулей.

В трансляторах средства поиска не предусмотрены. Если в задании требуется такая редкая работа как транслирование старого текста без предварительных изменений, то пользователь должен задавать специальное управляющее предложение корректору XUPDATE в виде

./␣RES␣NAME=<имя текста>

Когда в процессе просмотра библиотек обнаруживается, что некоторые нужные программы не присутствуют в библиотеке или среди управляющих предложений корректора имеются карты RES, то включается восстановительная программа XYKRES. Согласно переданному ей списку имен оператору ЭВМ выдается следующее сообщение:

XYK16I PROC RES TO <имя библиотеки>
XYK17I MEMBERS <список имен>
XYK11R DEFINE INPUT TAPE - VOL,DSN,LBL

В ответ оператор сообщает три координаты архивного файла - имя ленты, имя набора данных и порядковый номер набора данных. В ходе поиска восстанавливаемых разделов в архивном файле можно оперативно вмешиваться в работу программы: ме-

нять с пульта оператора список (добавить или удалить имена), спросить место нахождения, перемотать ленту в двух направлениях на заданное число блоков и т.д. При записи раздела в библиотеку захватывается ресурс, а по окончании записи оператору ЭВМ выдается сообщение

XYK15I TEXT (MODULE) <имя> <дата раздела> RESTORED

Программу XYKRES можно запустить и процедурой с пульта оператора – тогда необходимо указать имя библиотеки и список восстанавливаемых разделов. Отметим еще, что для функционирования системы автоматического восстановления приходится генерировать супервизор операционной системы так, чтобы все периферийные устройства считались неавтономными, несмотря на истинное положение.

В настоящее время в ВЦ ТГУ внедряется новая, более совершенная архивная система. Основными отличиями ее являются нестандартное представление архивных файлов на магнитных лентах и существование каталога архива на устройстве прямого доступа. Для работы с архивными файлами созданы собственные канальные программы и макрокоманды, что позволило значительно сократить время поиска нужных разделов. Применение каталога уменьшает ручной труд при обслуживании архива и дает возможность рассматривать и обработать архивные файлы на магнитных лентах подобными же методами что и библиотеки на устройствах прямого доступа.

# РАЗМЫТЫЙ ПОИСК ДОКУМЕНТАЛЬНОЙ ИНФОРМАЦИИ

К.А. Ээремаа

1. Автоматизированный поиск документов, соответствующих информационной потребности некоторого специалиста, является особым видом обработки данных. Основные трудности при решении этой проблемы связаны с формализацией описания содержания документов и с автоматизацией процесса составления таких описаний.

Проблема документального информационного поиска традиционно ставится следующим образом. Полагаются заданными множество документов $D = \{d_1, d_2, \ldots, d_m\}$ некоторой предметной области и множество существующих в этой области терминологических связей $\mathcal{R}$. Требуется найти функцию $F(D, \mathcal{R}, q)$, сопоставляющую с каждым запросом $q \in Q$ релевантные ему документы, т.е. ищется функция

$$F : Q \to 2^D,$$

где $2^D$ – множество всех подмножеств документов. Целью указания $\mathcal{R}$ в качестве аргумента функции F является подчеркивание возможности использовать в процессе поиска знания, не почерпнутые непосредственно из документов. Обычно $\mathcal{R}$ называется тезаурусом и содержит разные семантические связи между тер-

минами рассматриваемой области.

Для того, чтобы избежать непосредственного сравнения запроса с документами, каждый документ в информационно-поисковой системе (ИПС) заменяется его поисковым образом, передающим в формализованном виде основную тематику документа. Для этого вводится множество поисковых признаков (индексов) $X = \{x_1, x_2, \ldots, x_n\}$, являющихся в общем случае терминами соответствующей предметной области, и поисковым образом документа (ПОД) считается множество содержащихся в $d \in D$ терминов: $\delta = \{x : x < d\}$. Тогда функцию F можно рассматривать как суперпозицию двух функций $F_1(D, \mathcal{R}, X)$ и $F_2(\Delta, \mathcal{R}, q)$, где

$$F_1 : D \to \Delta, \quad F_2 : Q \to 2^D$$

и $\Delta = \{\delta_1, \delta_2, \ldots, \delta_m\}$ – множество поисковых образов документов.

Множество терминологических отношений $\mathcal{R}$ считается заданным в виде бинарного отношения, определенного на множестве $X \times X$:

$$\mathcal{R} \equiv \mathcal{R}(X, X) = \{(x, y) : x, y \in X\}.$$

То, что $\mathcal{R}$ является аргументом как функции индексирования $F_1$, так и функции выработки ответа $F_2$, указывает на возможность применения дополнительных знаний (или некоторых их частей) в обоих процессах, с одной стороны для более адекватного формального описания содержания документов и с другой – для более точного формирования множества ответных документов.

Основным недостатком традиционных ИПС, построенных на базе теории множеств, является ограниченность *применяемой в*

них двухвалентной логики. Ограниченный запас придаваемых разным утверждениям значений (истинно, ложно) создает семантического рода трудности в описании работы ИПС. Например, с точки зрения информационного поиска, индексы документа имеют разные степени существенности, которые должны быть учтены при определении релевантных документов. Трудности возникают и при применении тезауруса: не ясно, какие виды связей и с какой глубиной использовать, и являются ли порожденные посредством тезауруса индексы вообще равносильными с выбранными непосредственно из документа. Аналогично обстоит дело и при составлении запроса: хотя клиент знает, что некоторые из включенных им в запрос терминов являются более существенными чем другие, у него нет прямой возможности это указать. Правда, в какой-то мере недостатки такой системы компенсируются сложными методами сравнения запроса с ПОД, но полностью это всех недостатков не устраняет.

По указанным причинам мы в дальнейшем отказываемся от двухвалентной логики и допускаем, что высказывания могут принимать значения из фиксированного интервала $[0,1]$. При этом будем опираться на следующие принципы:

$1^{\circ}$ при любых $x \in X$ и $d \in D$ можно найти число $\varphi(d,x) \in [0,1]$, характеризующее значимость применения $x$ в качестве индекса документа $d$;

$2^{\circ}$ для любого $(x,y) \in \mathcal{R}$ можно найти число $r(x,y) \in [0,1]$, характеризующее значимость использования этой связи в ИПС.

Ставим себе целью определить правило, по которому при любых $q \in Q$ и $d \in D$ можно найти число $h(d,q) \in [0,1]$, характеризующее степень релевантности документа $d$ по отношению запроса $q$.

Существенным при описании работы ИПС с такой системой весов является методика получения исходных оценок для $\varphi$ и $r$, а также ход их трансформирования в оценки $h$ о релевантности выдаваемых документов. При этом процесс трансформирования должен соответствовать нашим интуитивным представлениям об изменении существенности тех или иных утверждений.

Среди математических методов поставленным требованиям лучше всего удовлетворяет теория размытых множеств и отношений, так как основная идея этой теории заключается в том, что принадлежность элемента в некоторую совокупность характеризуется функцией принадлежности, принимающей значения из интервала [0,1]. Применению этой теории для описания информационного поиска до сих пор посвящено скромное число работ. Основное внимание в них уделяется методам классификации поисковых образов (см. напр. [1, 2, 6, 7]) и определению релевантных данному запросу документов (см. [3, 4, 5]). В настоящей статье главным образом рассматривается процесс автоматического индексирования в размытой ИПС с учетом семантических связей между терминами, а также образованием словосочетаний и устранением многозначности. Изложенный в конце статьи процесс выработки ответа служить лишь примером возможности использования тезауруса вместо индексирования в расширении запросов, сохраняя его прежнее влияние на работу ИПС.

2. Процесс индексирования начинается с анализа текста документа, в ходе которого составляется исходный ПОД, т.е. множество терминов, выражающих основную тематику документа. Затем, пользуясь общими знаниями о предметной области, ин-

дексатор дополняет полученный ПОД терминами, указывающими на связанные тематические области, но не примененные в документе непосредственно. При автоматическом индексировании дополнительные знания о предметной области должны быть оформлены определенным образом и зафиксирована схема их использования.

Начинаем с описания расширения исходного ПОД, а его составление проанализируем ниже.

Согласно первому принципу связь между множествами D и X можно рассматривать как размытое отношение на $D \times X$. По сути дела такая связь может быть задана несколькими отношениями, задающими разные способы определения принадлежащих к документу терминов. Определяем одно из таких отношений как

$$S \equiv S(D,X) = \left\{ (d_j, x_i) \mid f^o(d_j, x_i) \,:\, i=1,\dots,n;\ j=1,\dots,m \right\},$$

где $f^o(d_j, x_i) \in [0,1]$ и $f^o(d_j, x_i) \neq 0$ означает, что $x_i$ входит в документ $d_j$ (выше это было обозначено через $x_i \prec d_j$).

Размытое множество

$$\delta_j(X) = \left\{ x_i \mid f^o(d_j, x_i) : i=1,\dots,n \right\}$$

называем исходным поисковым образом документа $d_j$ на базе X. Множество всех исходных ПОД обозначаем через

$$\Delta = \Delta(X) = \left\{ \delta_1, \delta_2, \dots, \delta_m \right\}.$$

Термин $x \in X$ называем (исходным) индексом документа $d_j$, если его вес в (исходном) ПОД отличен от нуля.

Множество знаний о предметной области полагаем заданным в виде бинарного отношения $\mathcal{R}$, из которого выделяем называемое тезаурусом подмножество парадигматических видов связей

$$R \equiv R(X,X) = \{R_1, R_2, \ldots, R_l\}.$$

Входящими в тезаурус видами связей являются, например, целое–часть, род–вид и т.д. Согласно второму принципу каждое отношение $R_k \in R$ выражается в виде размытого отношения

$$R_k = \{(x,y) \mid r_k(x,y) : x,y \in X\}.$$

Кроме того, имея в виду дальнейшее применение, требуем, чтобы каждое отношение было рефлексивным, т.е. для любого $x \in X$ и $k=1,\ldots,l$ имеет место $r_k(X,X) = 1$.

Весы элементов тезауруса считаем заданными вне системы, так как к настоящему времени не существует алгоритма автоматического составления тезауруса с семантическими связями между терминами.

Определяем расширение исходного ПОД посредством фиксированного $R_k$ как произведение размытого множества на размытое отношение:

$$\delta_j^1(XR_k) = \delta_j(X) \cdot R_k = \{x_i \mid f_k^1(d_j, x_i) : i=1,\ldots,n\},$$

где

$$f_k^1(d_j, x_i) = \max_{1 \le t \le n} (f^0(d_j, x_t) \cdot r_k(x_t, x_i)).$$

Содержательно это означает, что $\delta_j^1(XR_k)$ состоит из тех терминов $y \in X$, которые посредством $R_k$ связаны с некоторым индексом $x \in \delta_j(X)$. Так как $f^0(d_j,x) \cdot r_k(x,y) \le f^0(d_j,x)$, то вес добавляемого термина (нового индекса) не превышает веса вызвавшего его индекса. Тем самым в процессе расширения исходного ПОД имеется тенденция уменьшения значимости получен-

ных "косвенным образом" индексов. Ввиду рефлексивности отношения $R_k$ вес исходного индекса не может уменьшаться, а может возрастать только в том случае, если он окажется вызванным и некоторым другим исходным индексом. Согласно сказанному, между $\delta_j(X)$ и $\delta_j^1(XR_k)$ имеет место включение

$$\delta_j(X) \subseteq \delta_j^1(XR_k).$$

На полученное расширение $\delta_j^1(XR_k)$ можно снова применить отношение $R_k$. В общем случае получаем формулу

$$\delta_j^s(XR_k) = \delta_j^{s-1}(XR_k) \cdot R_k,$$

где $s=1,2,\ldots$ и $\delta_j^0(XR_k) = \delta_j(X)$.

Можно показать (см. [8]), что приведенное расширение поискового образа обладает следующими свойствами.

$1^0$ $\delta_j^{s-1}(XR_k) \subseteq \delta_j^s(XR_k)$ для любого $s=2,3,\ldots$, т.е. каждый последующий ПОД шире предыдущих.

$2^0$ $\delta_j^s(XR_k) = \delta_j(X) \cdot R_k^s$, где $R_k^s = R_k \circ R_k \circ \ldots \circ R_k$ является $s$-кратной композицией размытых отношений, которая для двух сомножителей определяется равенством

$$R_k \circ R_k = \left\{(x,y) \mid r_k^2(x,y) : x,y \in X\right\},$$

где

$$r_k^2(x,y) = \max_{1 \le i \le n} (r_k(x,x_i) \cdot r_k(x_i,y));$$

таким образом, нахождение расширения ПОД нужной степени приводится к произведению исходного ПОД на соответственно расширенное отношение.

3. Для любого $R_k \in R$ найдется значение $s_k$ такое, что $R_k^{s_k} = R_k^{s_k+1} = \ldots$ и следовательно $\delta_j^{s_k}(XR_k) = \delta_j^{s_k+1}(XR_k)$; т.е. процесс расширения исходного ПОД является конечным.

Отношение $R_k^{s_k}$ называем транзитивным замыканием размытого отношения $R_k$ и обозначим через $\bar{R}_k$. Тогда максимальное расширение $\bar{\delta}_j(XR_k)$ исходного ПОД $\delta_j(X)$ с помощью $R_k$ выражается формулой

$$\bar{\delta}_j(XR_k) = \delta_j(X) \cdot \bar{R}_k .$$

Расширяя $\delta_j(X)$ при помощи всех отношений $R_1, R_2, \ldots, R_l$ получаем последовательность максимальных расширений исходного ПОД

$$\bar{\delta}_j(XR_1), \bar{\delta}_j(XR_2), \ldots, \bar{\delta}_j(XR_l).$$

Сохранить все эти варианты поискового образа в базе данных ИПС нецелесообразно как по чисто техническим причинам, так и потому, что "рядовому пользователю" трудно разобраться в семантических тонкостях отдельных видов отношений. Поэтому постараемся найти форму соединения всех разновидностей поискового образа одного документа в одно целое.

Пока что мы на отношения $R_1, R_2, \ldots, R_l$ не наложили никаких требований, кроме рефлексивности. В дальнейшем будем предполагать выполненность следующих требований:

$1^o$ отношения $R_1, R_2, \ldots, R_l$ являются независимыми в том смысле, что расширение ПОД по одному из них не влияет на расширения по остальным;

$2^o$ значимости применения отношений $R_1, R_2, \ldots, R_l$ в информационном поиске сравнимы между собой и могут быть выражены

коэффициентами $\lambda_1, \lambda_2, \ldots, \lambda_l \in (0,1]$.

Возвращаясь теперь к проблеме создания общего поискового образа, по этим требованиям самым логичным кажется определить его суммой

$$\bar{\delta}_j(XR) = \lambda_1 \cdot \bar{\delta}_j(XR_1) \cup \lambda_2 \cdot \bar{\delta}_j(XR_2) \cup \ldots \cup \lambda_l \cdot \bar{\delta}_j(XR_l) .$$

Легко показать, что

$$\bigcup_{k=1}^{l} \lambda_k \cdot \bar{\delta}_j(XR_k) = \delta_j(X) \cdot \bigcup_{k=1}^{l} \lambda_k \cdot \bar{R}_k .$$

Обозначая теперь сумму в правой стороне равенства через $\bar{R} \equiv \bar{R}(X,X)$, максимальное расширение исходного ПОД можно задать простой формулой

$$\bar{\delta}_j(XR) = \delta(X) \cdot \bar{R} .$$

4. Выдвинутое в предыдущем пункте требование независимости отношений тезауруса не выполнено для любых отношений. Примером такого отношения является синонимия, которую следует учитывать при применении остальных видов. В общем случае, при зависимых отношениях следует рассматривать расширения в виде произведения

$$\delta_j(X) \cdot R_{k_1}^{t_1} \cdot R_{k_2}^{t_2} \cdot \ldots \cdot R_{k_i}^{t_i},$$

где $k_\mu \in \{1, \ldots, l\}$, $t_\mu \in \{1, \ldots, s_{k_\mu}\}$. Это значит, что на расширенный с помощью некоторого отношения поисковый образ разрешается применять отношение другого вида. Рассмотрение таких смешанных расширений имеет смысл только в той мере, в какой им можно дать содержательную трактовку. Ниже проанали-

зируем случай, когда некоторый вид отношения, обозначаем его через $R_0$, определяет классы терминов, элементы которых в установленных ситуациях следует рассматривать взаимозаменяемыми. Тот факт, что термин является представителем такого класса терминов, следует учитывать в зависимости от семантики связи либо при осмыслении других видов отношений, либо при интерпретировании поискового образа. Аналогом таких классов в традиционных ИПС являются дескрипторные классы.

Итак, пусть в тезаурусе $\{R_0, R\}$ отношение

$$R_0 = \{(x,y) \mid r_0(x,y) : x,y \in X\}$$

является симметричным отношением (т.е. $r_0(x,y) = r_0(y,x)$), которое не удовлетворяет требованию независимости по отношению к остальным. Ввиду симметричности транзитивное замыкание $\bar{R}_0$ этого отношения определяет разбиение множества терминов X на классы связанных между собой элементов. Называем эти классы дескрипторными классами.

Полагаем теперь, что $R_0$ является отношением, создаваемые которым дескрипторные классы требуется учитывать только при трактовке расширенного ПОД, где любой индекс следует рассматривать как представитель соответствующего класса. Такая ситуация имеет место, например, тогда, когда $R_0$ задает ассоциативную связь между терминами. Очевидно, что в этом случае вместо поискового образа $\bar{\delta}_j(XR)$ следует пользоваться поисковым образом

$$\tilde{\delta}_j(XRR_0) = \bar{\delta}_j(XR)\cdot\bar{R}_0 = (\delta_j(X)\cdot\bar{R})\cdot\bar{R}_0 = \delta_j(X)\cdot(\bar{R}\circ\bar{R}_0) = \delta_j(X)\cdot\bar{R}\,.$$

Таким образом, каждый индекс из $\delta_j(XR)$ заменяется множеством

связанных с ним терминов, вес которых вычисляется по правилу произведения и зависит от веса первоначального термина и весов связей в дескрипторном классе.

Пусть теперь $R'_0$ является отношением дескрипторных классов, влияющим на остальные $R_k \in R$ таким образом, что любую пару $(x,y) \in R_k$ следует рассматривать как связь между представленными терминами x и y дескрипторными классами (например, если $R_k$ задает синонимию). Требуется найти максимальное расширение исходного ПОД с помощью $\{R'_0, R\}$.

Согласно условиям задачи на каждом шагу расширения индексы следует заменять на соответствующие классы. Таким образом, максимальное расширение $\tilde{\tilde{\delta}}_j(XR_kR'_0)$ выражается в виде произведения

$$\tilde{\tilde{\delta}}_j(XR_kR'_0) = \delta_j(X) \cdot (\overline{R_k \circ R'_0}),$$

где $\overline{R_k \circ R'_0}$ является транзитивным замыканием композиции $R_k \circ R'_0$. Учитывая все заданные в R отношения, получаем максимальное расширение в виде выражения

$$\tilde{\tilde{\delta}}_j(XRR'_0) = \bigcup_{k=1}^{l} \lambda_k \cdot \delta_j(X) \cdot (\overline{R_k \circ R'_0}) = \delta_j(X) \cdot \bigcup_{k=1}^{l} \lambda_k \cdot (\overline{R_k \circ R'_0}) = \delta_j(X) \cdot \tilde{\tilde{R}},$$

по внешнему виду аналогичного полученной выше формуле.

Во всех проанализированных случаях процесс расширения приводится к произведению $\delta_j(X) \cdot \hat{R}$, где $\hat{R}$ – обобщенный поисковый тезаурус, форма которого определяется по характеру семантических связей в тезаурусе, т.е. является отношением $\bar{R}$, $\tilde{R}$ или $\tilde{\tilde{R}}$.

Расширение $\hat{S}$ всей совокупности исходных связей S(D,X) при помощи обобщенного поискового тезауруса можно задать в

виде композиции

$$\hat{S}(D,X) = S(D,X) \circ \hat{R}(X,X) =$$

$$= \left\{(d_j,x_i) \mid \hat{f}(d_j,x_i) : i=1,\dots,n; j=1,\dots,m\right\}, \qquad (1)$$

где $\hat{f}(d_j,x_i) = \max\limits_{1\le t\le n} (f^0(d_j,x_t)\cdot\hat{f}(x_t,x_i))$, а $\hat{f}(x_t,x_i)$ является весовой функцией отношения $\hat{R}$.

5. Расширению исходного ПОД предшествует его составление по тексту документа. Критерием выбора исходных индексов считаем входимость соответствующих терминов в текст документа. Непосредственное автоматическое установление присутствия термина $x \in X$ в документе $d_j$ затруднено тем, что $x$ может быть выражен словосочетанием или омонимом и выступает в различных грамматических формах. Так как в тезаурусе рассматриваются семантические связи, то нельзя ограничиваться включением в $X$ лишь однословных и неомонимных понятий. Трудности, связанные с составлением $\delta_j(X)$, возникают в первую очередь при автоматизировании индексирования. Для их преодолевания будем строить $\delta_j(X)$ при помощи многошагового процесса: сначала выявляем совокупность входящих в документ однословных понятий (термов), а затем на основе этой совокупности и некоторых вспомогательных связей образуем словосочетания и устраняем многозначность.

Пусть заданы множество термов

$$Z = \left\{z_1,z_2,\dots,z_u\right\}$$

и множество понятий

$$Y = \left\{y_1,y_2,\dots,y_v\right\},$$

где терм $z \in Z$ является словом, имеющим самостоятельное терминологическое значение или же входящим в состав некоторого словосочетания, а понятие $y \in Y$ – это слово или словосочетание. Словарь Y отличается от X только тем, что в X различные значения омонима $y \in Y$ обозначаются специальными словами. Словарь Z не содержит термов, не являющихся либо понятиями либо частями понятий.

Любое понятие $y_k \in Y$ представимо в виде последовательности составляющих его термов

$$y_k \rightarrow z_1^k \sqcup z_2^k \sqcup \ldots \sqcup z_\mu^k ,$$

которая в случае однословного понятия состоит только из одного элемента. Однако, так как в зависимости от построения предложения документа составляющие понятие термы могут встречаться почти в любой последовательности, то с каждым понятием $y_k$ сопоставим множество

$$y_k \rightarrow \left\{ z_1^k, z_2^k, \ldots, z_\mu^k \right\}.$$

Тем самым на множестве $Z \times Y$ установлено отношение

$$A(Z,Y) = \left\{ (z,y) \mid \alpha(z,y) : z \in Z, y \in Y \right\},$$

где $\alpha(z,y) = 1$, если z входит в множество термов понятия y.

Если игнорировать грамматические формы слов текста документа, как это делается в большинстве документальных ИПС, то установление входимости $z < d$ легко автоматизируемо. Соответствующий процесс в настоящей статье не рассматривается, одна возможность его автоматизирования приведена в [7]. В данном случае полагаем, что задана процедура $P(d_j, Z)$, устанавливаю-

щая для любого $z \in Z$ его входимость или невходимость в $d_j$ с приданием соответствующего веса (для определения веса можно использовать разные статистические частотные оценки).

Итак, в результате работы процедуры $P(d_j,Z)$ получаем размытое множество

$$\delta_j(Z) = \{ z \mid f_Z(d_j,z) : z \in Z \},$$

где $f_Z(d_j,z) \neq 0$, если $z \prec d_j$. Кроме того допустим, что $P(d_j,Z)$ группирует термы $z$ в $\delta_j(Z)$ по предложениям, т.е. в $\delta_j(Z)$ сохраняется структура предложений документа $d_j$:

$$\delta_j(Z) = \{ \delta_j^{(1)}(Z), \delta_j^{(2)}(Z), \ldots, \delta_j^{(t_j)}(Z) \},$$

где $\delta_j^{(\tau)}(Z)$ при $1 \leq \tau \leq t_j$ является поисковым образом $\tau$-ого предложения, а $t_j$ – числом предложений документа $d_j$. Каждый поисковый образ предложения задается в виде размытого множества

$$\delta_j^{(\tau)}(Z) = \{ z \mid f_Z^{(\tau)}(d_j,z) : z \in Z \},$$

где $f_Z^{(\tau)}(d_j,z) = f_Z(d_j,z)$, если $z$ содержится в $\tau$-ом предложении.

Проанализируем теперь переход от $\delta_j(Z)$ на поисковый образ $\delta_j(Y)$. Скажем, что понятие $y_k$ содержится в документе $d_j$, если все составляющие его термы $z_1^k, z_2^k, \ldots, z_\mu^k$ входят в одно и то же предложение документа. Исходя из этого соображения с помощью отношения $A(Z,Y)$ и поискового образа $\delta_j^{(\tau)}(Z)$ можно найти поисковый образ $\tau$-ого предложения на базе понятий как размытое множество

$$\delta_j^{(\tau)}(Y) = \left\{ y \mid f_Y^{(\tau)}(d_j, y) : y \in Y \right\},$$

если положить

$$f_Y^{(\tau)}(d_j, y) = \min_{1 \le l \le u} \left( \max \left( f_Z^{(\tau)}(d_j, z_l), 1 - \alpha(z_l, y) \right) \right).$$

Легко проверить, что действительно $f_Y^{(\tau)}(d_j, y_k) \neq 0$ тогда и только тогда, когда

$$f_Z^{(\tau)}(d_j, z_1^k), f_Z^{(\tau)}(d_j, z_2^k), \ldots, f_Z^{(\tau)}(d_j, z_\mu^k) \neq 0,$$

т.е. если $\tau$-ое предложение содержит все термы, составляющие понятие $y_k$. При этом вес понятия $y_k$ в этом предложении приравнивается наименьшему из весов соответствующих термов. Переход от поисковых образов предложений на поисковый образ документа определяем суммой

$$\delta_j(Y) = \bigcup_{\tau=1}^{t_j} \delta_j^{(\tau)}(Y).$$

Для получения исходного поискового образа $\delta_j(X)$ потребуется еще устранить из $\delta_j(Y)$ неоднозначность, обусловленную возможным применением омонимов. В случае любого омонимного понятия $y_k$ известно множество его возможных значений $\left\{ x_1^k, x_2^k, \ldots, x_\nu^k \right\}$, которое для неомонимного понятия состоит из самого этого понятия как термина. Для выбора в качестве индекса подходящего значения связываем с каждым термином его возможный контекст в виде перечня понятий, появление которых в документе с некоторой оценкой указывает на определенное значение. Таким образом, полагаем заданными размытые отношения

$$B_1(Y,X) = \left\{ (y,x) \mid \beta_1(y,x) : y \in Y,\ x \in X \right\},$$

$$B_2(Y,X) = \left\{(y,x) \mid \beta_2(y,x) : y \in Y,\ x \in X\right\},$$

где $\beta_1(y,x) = 1$ тогда и только тогда, когда $x$ является значением (омонимного) понятия $y$ и $\beta_2(y,x) \neq 0$ только тогда, когда $y$ входит в контекст термина $x$.

Исходный поисковый образ $\delta_j(X)$ определяется теперь через $\delta_j(Y)$, $B_1(Y,X)$ и $B_2(Y,X)$ формулой

$$\delta_j(X) = \delta_j(Y) \cdot B_1(Y,X) \cap \delta_j(Y) \cdot B_2(Y,X).$$

В полученном таким образом исходном ПОД значение $x_\eta^k$ омонима $y_k$ является исходным индексом тогда и только тогда, когда $y_k \prec d_j$ и в $d_j$ содержится по меньшей мере одно понятие из контекста термина $x_\eta^k$. Возможен и случай, когда документ индексируется по нескольким значениям, если контексты соответствующих терминов пересекаются, но предпочтение тому или другому термину дает в этой ситуации вес.

6. Обратимся теперь к описанию процесса выработки ответа. При этом мы не ставим себе целью всестороннее описание процесса, а лишь уточнение роли тезауруса. Покажем, каким образом тезаурус может быть использован в процессе выработки ответа вместо процесса индексирования, чтобы сохранялось его прежнее влияние на информационный поиск.

Следуя идеологии построенной модели ИПС, информационный запрос должен включать фактор размытости. С точки зрения пользователя это означает, что ему разрешается оценивать степени существенности примененных в запросе терминов. Весы терминов должны влиять на весы выделенных документов и тем самым выдвигать те или иные из них.

Структуру информационного запроса, а также и процедуру нахождения ответа, можно задавать многими способами в зависимости от специфики предметной области и подготовленности пользователей. Приведем здесь один из простейших возможностей, где сначала вводим понятие элементарного запроса, а более сложный вариант получается комбинированием элементарных.

Назовем размытое множество

$$q(X) = \left\{ x \mid g(x) : x \in X \right\}$$

элементарным запросом, на который требуется найти ответ на базе составленного выше отношения $\hat{S}(D,X)$. Под ответом понимаем множество документов

$$a(D) = \left\{ d_j \mid h(d_j) : j=1,\ldots,m \right\},$$

в которых содержится по меньшей мере один из перечисленных в запросе терминов. При этом вес $h(d)$ должен с одной стороны зависеть от соответствующих весов терминов в $q(X)$ и с другой - в $\hat{S}(D,X)$.

Определяем ответ $a(D)$ на элементарный запрос $q(X)$ как размытое множество документов, вычисленное по формуле

$$a(D) = q(X) \cdot (\hat{S}(D,X))^{-1}, \qquad (2)$$

где

$$(\hat{S}(D,X))^{-1} = \hat{S}^*(X,D) =$$

$$= \left\{ (x_i,d_j) \mid \hat{f}^*(x_i,d_j) : i=1,\ldots,n; j=1,\ldots,m \right\}$$

и $f^*(x_i,d_j) = \hat{f}(d_j,x_i)$ обозначает результат обращения отношения $\hat{S}$. Согласно определению произведения размытого мно-

жества на размытое отношение получаем

$$h(d_j) = \max_{1 \le i \le n} (g(x_i) \cdot \hat{f}^*(x_i, d_j)).$$

Более сложные запросы можно составлять исходя из элементарных, указывая операции, проводимые между ответными множествами. Так, например, с запросом $q_1 \& q_2$ логично связать ответ $a_1 \cap a_2$, состоящий из тех документов, которые содержат по меньшей мере одну пару $(x,y)$, где $x \in q_1$ и $y \in q_2$. Аналогично можно определить еще целый ряд операций, заключающихся в действиях над ответами или же в определенном изменении весовой функции.

Преобразуем теперь формулу (2), применяя для этого легко доказуемое (см. [8]) равенство:

$$(S(D,X) \circ \hat{R}(X,X))^{-1} = \hat{R}^*(X,X) \circ S^*(X,D).$$

Учитывая формулу (1) получаем

$$\begin{aligned} a(D) &= q(X) \cdot (\hat{S}(D,X))^{-1} = \\ &= q(X) \cdot (S(D,X) \circ \hat{R}(X,X))^{-1} = \\ &= q(X) \cdot (\hat{R}^*(X,X) \circ S^*(X,D)) = \\ &= (q(X) \cdot \hat{R}^*(X,X)) \cdot S^*(X,D) , \end{aligned}$$

т.е. с точки зрения получения ответа расширение исходных ПОД с помощью тезауруса $\hat{R}$ равносильно расширению информационного запроса с обратным тезаурусом $\hat{R}^*$.

Значит, если в системе сохраняются исходные поисковые образы, то клиенту разрешается пользоваться своим тезаурусом и указанная возможность не требует перенастройки всей системы. Вполне вероятно, что такой тезаурус является лишь отно-

шением некоторого вида, конкретный состав которого уточняется в ходе диалога с системой посредством дисплея.

7. В конструированной модели размытой ИПС исходными данными являются следующие множества:

$D = \{d_1, d_2, \ldots, d_m\}$ – документы,

$Z = \{z_1, z_2, \ldots, z_u\}$ – термы,

$Y = \{y_1, y_2, \ldots, y_v\}$ – понятия,

$X = \{x_1, x_2, \ldots, x_n\}$ – индексы,

$\mathfrak{R} = \{R_0, R_1, \ldots, R_l, A(Z,Y), B_1(Y,X), B_2(Y,X)\}$ – терминологические отношения.

Взаимную зависимость этих множеств, с точки зрения их получения, характеризует приведенная на рис. 1 схема.

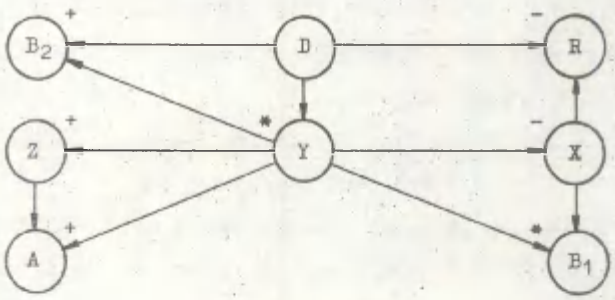



Рис. 1. Зависимость между множествами при их составлении.
Знак + означает, что составление множества автоматизируемо, * – полуавтоматизируемо, – – в основном неавтоматизируемо.

Оказывается, что составление большинства из приведенных множеств автоматизируемо. Самым трудоемким оказывается составление тезауруса, но и тут применимы некоторые методы автоматизирования, помогающие составителям при контроле логич-

ности и при систематизировании (см. [8]).

Что касается практической реализации размытой ИПС, то применяемые структуры данных ввиду своей простоты определимы в случае большинства баз данных.

## Литература


1. Negoita, C.V., On the Application of the Fuzzy Sets Separation Theorem for Automatic Classification in Information Retrieval Systems. Information Sciense, 1973, 5.
2. Negoita, C.V., On the Notation of Relevance in Information Retrieval. Kybernetes, 1973, 2, 3.
3. Radecki, T., Mathematical Model of Time-effective Information Retrieval System Based on the Theory of Fuzzy Sets. Inf. Proc. and Man., 1977, 13, 2.
4. Radecki, T., Fuzzy Sets Theoretical Approach to Document Retrieval. Inf. Proc. and Man., 1979, 15.
5. Sachs, W.M., An Approach to Associative Retrieval through the Theory of Fuzzy Sets. J. of the American Socicty for Inf. Science, 1976, 27, 2.
6. Реброва М.П., Размытые множества и теория классификации. НТИ, 1976, 2, 10.
7. Ээремаа К.А., Выделение наиболее существенных классов данных. Труды ВЦ ТГУ, 1980, 43.
8. Ээремаа К.А., Моделирование размытого документального информационного поиска. Диссертация (рукопись), Тарту, 1981.

С о д е р ж а н и е

ОБРАБОТКА ДАННЫХ НА ЕС ЭВМ.
Труды вычислительного центра.
Выпуск 47.
На русском языке.
Тартуский государственный университет.
ЭССР, 202 400, г.Тарту, ул.Юликооли, I8.
Ответственный редактор Ю. Тапфер.
Сдано в печать 6.11.1981.
МВ 09019.
Формат 30х42/4.
Бумага писчая.
Машинопись. Ротапринт.
Условно-печатных листов 3,49.
Учетно-издательских листов 2,35.
Печатных листов 3,75.
Тираж 200.
Заказ № I226.
Цена 35 коп.
Типография ТГУ, ЭССР, 202400, г.Тарту, ул.Пялсона, I4.