Описание функций библиотеки IRBIS64_CLIENT.DLL

1. Общие замечания

Библиотека irbis64_client.dll предназначена для полнофункционального доступа к базам данных ИРБИС64 на основе клиент-серверной архитектуры. (т.е. путем взаимодействия с TCP/IP сервером ИРБИС64).

В функциях используются два типа данных: целые числа (integer) и строковые данные (Pchar).

Все строковые данные передаются и возвращаются в кодировке UTF8 – кроме случаев, когда оговаривается иное.

Все строковые данные возвращаются функциями через буфера (тип Pchar), память для которых должна выделяться в вызываемой программе.

Строковые данные представляют собой последовательность байт (тип char) с двоичным нулем в конце. Строковые данные могут разбиваться на отдельные строки с помощью последовательности двух байт - $0D0A.

Прототипы функций, коды возврата и используемые константы описаны в файле irbis64_client.pas

Все функции могут завершаться – помимо специфических – следующими кодами возврата:

ERR_USER – функция прервана пользователем или иная ошибка общего характера;

ERR_BUSY - не завершена обработка предыдущего запроса (функции);

ERR_UNKNOWN - неизвестная ошибка;

ERR_BUFSIZE – размер выходного буфера недостаточен для возвращаемых данных.

2. Функции общего назначения

2.1 Регистрация клиента на сервере

```
function IC_reg(aserver_host: Pchar;
        aserver_port: Pchar;
        arm:char;
        user_name,password: Pchar;
        var answer: Pchar; abufsize: integer):integer;
```

Исходные данные:

aserver_host – адрес сервера в числовом виде (например 192.168.5.140).

aserver_port – рабочий порт сервера (6666).

arm – тип клиента; принимает значения:

IRBIS_READER – АРМ Читатель;

IRBIS_CATALOG – АРМ Каталогизатор;

IRBIS_COMPLECT – АРМ Комплектатор;

IRBIS_BOOKLAND – АРМ Книговыдача;

IRBIS_ADMINISTRATOR – АРМ Администратор;

IRBIS_BOOKPROVD – АРМ Книгообеспеченность.

user_name – имя пользователя, зарегистрированного на сервере.
password – пароль пользователя.
answer – выходной буфер для возвращаемых данных.
abufsize - размер выходного буфера.

Возвращаемые данные:
Код возврата – принимает следующие значения:

ZERO – успешное завершение функции;

CLIENT_NOT_IN_LIST - указанный пользователь не зарегистрирован;

WRONG_PASSWORD – указан неверный пароль;

CLIENT_LIST_OVERLOAD – превышено максимальное кол-во текущих пользователей **;**

CLIENT_NOT_ALLOWED – указанный пользователь не имеет доступа к указанному АРМу.

В случае успешного выполнения функции (регистрации) в выходном буфере (answer) возвращается профиль пользователя. Профиль пользователя представляет собой набор строк (т.е. данных, разделенных символами $0D0A) и имеет структуру INI-файла, который определен в учетной записи соответствующего пользователя на сервере. Данные возвращаются в ANSI-кодировке.

Комментарий:
Данная функция должна обязательно выполняться первой в клиентском приложении (ей могут предшествовать лишь функции установки общих параметров – см. далее).

2.2 Раз-регистрация клиента на сервере (сигнал об окончании работы)

function IC_unreg(user_name: Pchar): integer;

Исходные данные:
user_name – имя пользователя.

Возвращаемые данные:
Код возврата – принимает следующие значения:

ZERO – успешное завершение функции;

Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Функцию необходимо выполнять в конце работы клиентского приложения. Сервер производит автоматическую раз-регистрацию клиентов, не подающих запросов в течении определенного времени – см. функцию IC_set_client_time_live.

2.3 Установка интервала подтверждения

function IC_set_client_time_live(Aopt: integer): integer;
Исходные данные:
Aopt – интервал времени в минутах.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Функция определяет интервал времени, по истечении которого автоматически выдается подтверждение о том, что клиентское приложение находится в рабочем состоянии («живет») – если в течение этого интервала не выдавался ни один запрос на сервер. По умолчанию – 1 минута. Автоматическое подтверждение выполняется с помощью функции IC_nooperation.

2.4 Установка времени появления заставки ожидания.

function IC_set_show_waiting(Aopt: integer): integer;

Исходные данные:
Aopt – интервал времени в секундах.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Функция определяет интервал ожидания ответа от сервера, по истечении которого появляется заставка с «бегущим ирбисом». По умолчанию – 3 секунды.

2.5 Установка режима работы через Web-шлюз

function IC_set_webserver(Aopt: integer): integer;

Исходные данные:
Aopt – принимает два значения:
1 – включить режим работы через Web-шлюз;
0 – отключить режим работы через Web-шлюз (по умолчанию).

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Подробнее о режиме работы с сервером ИРБИС64 через Web-шлюз смотри в описании сервера (Сервер 64.doc)

2.6 Установка имени шлюза при работе через Web-шлюз

function IC_set_webcgi(Acgi: Pchar): integer;

Исходные данные:
Acgi – имя шлюза (по умолчанию - /cgi-bin/wwwirbis.exe).

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

2.7 Установка режима ожидания ответа от сервера

function IC_set_blocksocket(Aopt: integer): integer;

Исходные данные:
Aopt – принимает два значения:
1 – включить режим «мертвого» ожидания ответа от сервера
0 – отключить режим «мертвого» ожидания ответа от сервера (по умолчанию).

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

2.8 Определение завершения очередного обращения к серверу

function IC_isbusy: integer;

Возвращаемые данные:
Код возврата – принимает два значения:
1 – обращение к серверу не завершено
0 – обращение к серверу завершено.

3. Функции работы с ресурсами

3.1 Обновление INI-файла – профиля пользователя на сервере

function IC_update_ini(inifile: Pchar): integer;

Исходные данные:
inifile – набор строк в виде структуры INI-файла (в ANSI-кодировке).

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

Комментарий:
В результате успешного выполнения функции обновляется (пополняется) INI-файл – профиль пользователя на сервере, в соответствии с которым выполнялась его регистрация (см. функцию IC_reg).

3.2 Чтение текстового ресурса (файла)

function IC_getresourse(Apath: integer; Adbn,Afilename: Pchar; var answer: Pchar; abufsize: integer): integer;

Исходные данные:
Apath – код, определяющий относительный путь размещения ресурса ИРБИС64; принимает следующие значения:
SYSPATH – общесистемный путь (т.е. там, где размещается собственно исполняемый модуль сервера \irbis64\);
DATAPATH – путь, где размещаются сведения о базах данных (\irbis64\datai\)$
DBNPATH2 – путь, где размещается собственно база данных;
DBNPATH3 - путь, где размещается собственно база данных;
DBNPATH10 - путь, где размещается собственно база данных.
Adbn – имя базы данных (не используется, если в качестве Apath указывается SYSPATH или DATAPATH);
Afilename – имя требуемого текстового файла (ресурса) с расширением;
answer – буфер, в котором возвращается требуемый ресурс;
abufsize – размер буфера для возвращаемых данных.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

При успешном завершении функции в **answer** возвращается требуемый текстовый ресурс. Данные возвращаются в ANSI-кодировке..

Комментарий:
Содержимое запрашиваемых ресурсов кэшируется в памяти IRBIS64_CLIENT.DLL.

3.3 Очистка памяти кэша

function IC_clearresourse: integer;

Исходные данные:
функция не имеет параметров

Возвращаемые данные:
Код возврата –ZERO

Комментарий:
В результате выполнения функции очищается кэш, в котором сохраняются запрошенные текстовые ресурсы (после чего при их запросе они берутся с сервера). При выполнении функции не производится обращение на сервер.

3.4 Групповое чтение текстовых ресурсов

function IC_getresoursegroup(var acontext: Pchar; var answer: Pchar; abufsize: integer): integer;

Исходные данные:
acontext – набор строк, каждая из которых соответствует одному запрашиваемому ресурсу и имеет вид:

Apath+'.'+Adbn+'.'+Afilename

Например: DBNPATH10.IBIS.BRIEF.PFT

О параметрах **Apath, Adbn, Afilename** см. в описании функции IC_getresourse;
answer – буфер для возвращаемых данных;
abufsize – размер буфера для возвращаемых данных.

Возвращаемые данные:
Код возврата – принимает следующие значения:

ZERO – успешное завершение функции;

Другие коды возврата свидетельствуют об ошибке.
После успешного выполнения функции в **acontext** сохраняются строки, соответствующие НЕ ПУСТЫМ ресурсам (число и порядок строк в **acontext** может измениться). В answer возвращается набор строк, каждая из которых соответствует строке в **acontext** и содержит соответствующий текстовый

ресурс – в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130 (Для подобных преобразований предлагаются специальные вспомогательные функции IC_reset_delim и IC_delim_reset).
Данные возвращаются в ANSI-кодировке.

Комментарий:
Содержимое запрашиваемых ресурсов кэшируется в памяти IRBIS64_CLIENT.DLL.

3.5 Чтение двоичного ресурса

function IC_getbinaryresourse(Apath: integer; Adbn,Afilename: Pchar; var Abuffer: PBuffer): integer;

Исходные данные:
Apath, Adbn, Afilename – аналогичны одноименным параметрам функции IC_getresourse;
Abuffer – указатель на специальный буфер для двоичного ресурса, тип которого описан в irbis64_client.pas. Память для этого буфера выделяется и освобождается в irbis64_client.dll.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.
После успешного выполнения функции сведения о двоичном ресурсе возвращаются в **Abuffer** (собственно двоичный ресурс в Abuffer^.data).

Комментарий:
Содержимое запрашиваемого ресурса кэшируется в памяти.

3.6 Запись текстового ресурса на сервер

function IC_putresourse(Apath: integer; Adbn,Afilename,Aresourse: Pchar): integer;

Исходные данные:
Apath, Adbn, Afilename – аналогичны одноименным параметрам функции IC_getresourse;
Aresourse – буфер, содержащий исходный текстовый ресурс (в ANSI-кодировке).

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;

Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Содержимое исходного ресурса кэшируется в памяти; если таковой уже запрашивался (находился в кэше) – он обновляется.

4. Функции для работы с файлом документов (мастер-файлом) базы данных

4.1 Чтение записи

function IC_read(Adbn: Pchar; Amfn,Alock: integer; var answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Amfn – номер записи (MFN);
Alock – параметр, принимающий два значения:
1 – блокировать запись при чтении
0 – читать запись без блокировки;
answer – буфер, в котором возвращается запрашиваемая запись;
abufsize – размер буфера для возвращаемых данных.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
REC_DELETE – запись логически удаленная;
REC_PHYS_DELETE – запись физически удаленная
READ_WRONG_MFN – несуществующий MFN;
При успешном завершении функции (код возврата ZERO, REC_DELETE, REC_PHYS_DELETE) в **answer** возвращается запись в клиентском представлении, а именно – как набор строк следующей структуры:
0#<код возврата>
MFN#<статус записи>
0#<номер версии записи>
далее следуют строки вида:
TAG#<значение поля>
где TAG – числовая метка поля.

Комментарий:
Для работы с записью в клиентском представлении (читать/редактировать данные) следует использовать **функции для работы с записью** (см. ниже раздел 5).

4.2 Чтение записи с расформатированием

function IC_readformat(Adbn: Pchar; Amfn,Alock: integer; Aformat: Pchar; var answer: Pchar; abufsize: integer; var answer1: Pchar; abufsize1: integer): integer;

Исходные данные:
Adbn, Amfn, Alock, answer, abufsize – аналогичны одноименным параметрам функции IC_read;
Aformat – непосредственный формат (на языке форматирования ИРБИС) или имя файла формата без расширения с предшествующим символом @ - например @brief, - в соответствии с которым будет расформатироваться документ;
answer1 – буфер для результата расформатирования;
abufsize1 – размер буфера для расформатирования.

Возвращаемые данные:
Аналогичны функции IC_read.
Дополнительно – в **answer1** возвращается результат расформатирования в виде набора строк, первая из которых содержит код результата расформатирования.

4.3 Запись/обновление записи в базе данных

function IC_update(Adbn: Pchar; Alock,Aifupdate: integer; var answer: Pchar; abufsize: integer): integer; stdcall;

Исходные данные:
Adbn – имя базы данных;
Alock – параметр принимает два значения:
1 – оставить запись заблокированной, если она была заблокирована;
0 – разблокировать запись, если она была заблокирована.
Aifupdate - параметр принимает два значения:
1 – актуализировать запись после записи/обновления;
0 – не актуализировать запись.
answer – буфер, содержащий исходную запись в клиентском представлении; в нем же будет возвращаться обновленная запись при успешном выполнении функции;
abufsize – размер буфера для возвращаемых данных.

Возвращаемые данные:
Код возврата – при успешном завершении функции число большее нуля и равное максимальному MFN в базе данных; в противном случае – код ошибки, в частности:

VERSION_ERROR – несовпадение номера версии записи (номер версии реальной записи в БД больше номера версии исходной записи); в этом случае **answer** будет содержать копию реальной записи из БД.
При успешном завершении функции в **answer** содержится обновленная запись. (Отличие исходной записи от возвращаемой может быть связано с результатом работы АВТОВВОДА – кроме того, если создается новая запись, в возвращаемой записи содержится MFN созданной записи.)

Комментарий:
Если MFN исходной записи является реальным (т.е. такой номер в базе данных есть), то соответствующая запись в БД полностью обновляется, в противном случае – в БД создается новая запись.

4.4 Групповая запись/обновление записей в базе данных

function IC_updategroup(Adbn: Pchar; Alock,Aifupdate: integer; var answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn, Alock, Aifupdate – аналогичны одноименным параметрам функции IC_update;
answer – буфер с исходными записями; каждая запись представляется в виде одной строки – которую необходимо сформировать из клиентского представления записи путем замены реальных разделителей полей $0D0A на псевдоразделители $3130; клиентское представление записи смотри в описании функции IC_read;
abufsize – размер буфера.

Возвращаемые данные:
Код возврата – при успешном завершении функции число большее нуля и равное максимальному MFN в базе данных; в противном случае – код ошибки.

Комментарий:
После выполнения функции записи в исходном буфере НЕ ОБНОВЛЯЮТСЯ! Обновляется только статус записи и MFN (в случае создания новой записи). Т.е. в случае необходимости повторного обновления записей, отредактированных этой функцией, их следует повторно прочитать.

4.5 Разблокировать запись

function IC_runlock(Adbn: Pchar; Amfn: integer): integer;

Исходные данные:
Adbn – имя базы данных;

Amfn – mfn записи.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Функция не применяется, если многопользовательский режим реализуется на основе версии записи.

4.6 Актуализировать запись

function IC_ifupdate(Adbn: Pchar; Amfn: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Amfn – mfn записи.

Возвращаемые данные:
Код возврата – принимает следующие значения:
ZERO – успешное завершение функции;
Другие коды возврата свидетельствуют об ошибке.

Комментарий:
Функция применяется в тех случаях, когда производилось редактирование записей (IC_update или IC_updategroup) без актуализации.

4.7 Получить максимальный MFN базы данных

f**unction IC_maxmfn(Adbn: Pchar): integer;**

Исходные данные:
Adbn – имя базы данных;

Возвращаемые данные:
Код возврата – при успешном завершении функции положительное число maxmfn+1, где maxmfn – максимальный mfn в базе данных.

4.8 Связанная групповая запись/обновление записей в базах данных

function IC_updategroup_sinhronize(Alock,Aifupdate: integer; Adbnames: Pchar; var answer: Pchar; abufsize: integer):integer;

Исходные данные:

Alock, Aifupdate, answer, abufsize – аналогичны одноименным параметрам функции IC_updategroup;
Adbnames – буфер со списком имен исходных баз данных, соответствующих исходным записям в answer, в виде набора строк.

Возвращаемые данные:
Код возврата – ZERO при успешном завершении функции или код ошибки.

Комментарий:
Данная функция отличается от .IC_updategroup тем, что, во-первых, исходные записи могут быть из разных баз данных, и во-вторых – редактирование записей происходит только в том случае, если этот процесс оказывается успешным для ВСЕХ исходных записей. Откорректированные записи не изменяются в исходном буфере.

5. Функции для работы с записью

Все функции данной группы предназначены для работы с записью в клиентском представлении (см. описание функции IC_read) и не связаны с обращением к серверу ИРБИС64, т.е. все изменения, выполняемые данными функциями, происходят в клиентском представлении записи, а не в реальной записи базы данных. Для реального обновления записи служат функции IC_update и IC_updategroup

5.1 Определить порядковый номер поля в записи

function IC_fieldn(Arecord: Pchar; Amet,Aocc: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении;
Amet – метка поля;
Aocc – номер повторения поля (начиная с 1).

Возвращаемые данные:
Код возврата – порядковый номер поля (начиная с 1) или отрицательное число, если такового поля нет в записи.

5.2 Прочитать значение заданного поля/подполя

function IC_field(Arecord: Pchar; nf: integer; delim: char; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении;
nf – порядковый номер поля (полученный с помощью функции IC_fieldn);

delim – односимвольный разделитель подполя (если задается $00, то выдается значение поля целиком);
answer – буфер для возвращаемого значения;
abufsize – размер буфера.

Возвращаемые данные:
Код возврата – ZERO или ERR_BUFSIZE, если мал выходной буфер, или ERR_USER, если поле/подполе отсутствует в записи;
answer – буфер со значением поля/подполя.

5.3 Добавить поле в запись

function IC_fldadd(Arecord: Pchar; Amet,nf: integer; pole: Pchar; abufsize: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена обновленная запись;
Amet – метка добавляемого поля;
nf – порядковый номер добавляемого поля; если задать 0 – поле будет добавлено последним по порядку;
pole – буфер со значением добавляемого поля;
abufsize – размер буфера Arecord для обновленной записи.

Возвращаемые данные:
Код возврата – ZERO или ERR_BUFSIZE, если мал выходной буфер;
Arecord – буфер с обновленной записью.

5.4 Заменить поле в записи

function IC_fldrep(Arecord: Pchar; nf: integer; pole: Pchar; abufsize: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена обновленная запись;
nf – порядковый номер заменяемого поля (полученный с помощью функции IC_fieldn);
pole – буфер с заменяющим значением поля; если указать пустое значение – соответствующее поле удаляется из записи;
abufsize – размер буфера Arecord для обновленной записи.

Возвращаемые данные:
Код возврата – ZERO или ERR_BUFSIZE, если мал выходной буфер, или ERR_USER, если заменяемое поле отсутствует в записи;

Arecord – буфер с обновленной записью.

5.5 Определить количество полей в записи

function IC_nfields(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении;

Возвращаемые данные:
Код возврата – количество полей в исходной записи.

5.6 Определить количество повторений поля с заданной меткой

function IC_nocc(Arecord: Pchar; Amet: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении;
Amet – метка поля.

Возвращаемые данные:
Код возврата – количество повторений заданного поля.

5.7 Определить метку поля с заданным порядковым номером

function IC_fldtag(Arecord: Pchar; nf: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении;
nf – порядковый номер поля.

Возвращаемые данные:
Код возврата – метка заданного поля или ERR_USER, если такового нет в записи.

5.8 Опустошить запись

function IC_fldempty(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена опустошенная запись;

Возвращаемые данные:
Код возврата – ZERO;

Arecord – буфер с опустошенной записью.

5.9 Изменить mfn записи

function IC_changemfn(Arecord: Pchar; newmfn: integer): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена измененная запись;
newmfn – новое значение mfn.

Возвращаемые данные:
Код возврата – ZERO;
Arecord – буфер с измененной записью.

5.10 Установить в статусе записи признак логической удаленностит

function IC_recdel(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена измененная запись;

Возвращаемые данные:
Код возврата – ZERO;
Arecord – буфер с измененной записью.

5.11 Снять в статусе записи признак логической удаленности

function IC_recundel(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена измененная запись;

Возвращаемые данные:
Код возврата – ZERO;
Arecord – буфер с измененной записью.

5.12 Снять в статусе записи признак заблокированности

function IC_recunlock(Arecord: Pchar): integer;

Исходные данные:

Arecord – буфер с исходной записью в клиентском представлении; в этот же буфер будет помещена измененная запись;

Возвращаемые данные:
Код возврата – ZERO;
Arecord – буфер с измененной записью.

5.13 Прочитать mfn записи

function IC_getmfn(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении.

Возвращаемые данные:
Код возврата – mfn исходной записи;

5.14 Создать пустую запись

function IC_recdummy(Arecord: Pchar; abufsize: integer): integer;

Исходные данные:
Arecord – буфер, в котором будет создаваться пустая запись в клиентском представлении;
abufsize – размер буфера.

Возвращаемые данные:
Код возврата –ZERO или ERR_BUFSIZE, если мал буфер;
Arecord – буфер с пустой записью.

5.15 Прочитать в статусе записи признак актуализированности.

function IC_isActualized(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении.

Возвращаемые данные:
Код возврата – 1 - если запись актуализирована, 0 – если не актуализирована.

5.16 Прочитать в статусе записи признак заблокированности

function IC_isLocked(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении.

Возвращаемые данные:
Код возврата – 1 - если запись заблокирована, 0 – если не заблокирована.

5.17 Прочитать в статусе записи признак логической удаленности

function IC_isDeleted(Arecord: Pchar): integer;

Исходные данные:
Arecord – буфер с исходной записью в клиентском представлении.

Возвращаемые данные:
Код возврата – 1 - если запись логически удалена, 0 – если не удалена.

6. Функции для работы со словарем базы данных

6.1 Получить список терминов словаря, начиная с заданного

function IC_nexttrm(Adbn,Aterm: Pchar; Anumb: integer; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Aterm – исходный термин;
Anumb – количество запрашиваемых терминов; если задается 0 – будут выдаваться все термины до конца словаря, но не более, чем MAX_POSTINGS_IN_PACKET;
answer – буфер для списка возвращаемых терминов;
abufsize – размер буфера.

Возвращаемые данные:
Код возврата – может принимать следующие значения:

ZERO – исходный термин найден в словаре;

TERM_NOT_EXISTS – исходный термин не найден в словаре, при этом возвращается Anumb следующих (ближайших) терминов;

TERM_LAST_IN_LIST – исходный термин больше последнего термина в словаре; при этом возвращается последний термин словаря;

TERM_FIRST_IN_LIST – исходный термин меньше первого термина в словаре; при этом возвращается Anumb первых терминов словаря;

ERR_BUFSIZE – выходной буфер мал для запрашиваемых терминов; при этом ничего в выходном буфере не возвращается.

answer – буфер со списком терминов в виде набора строк следующей структуры:

nnn#<термин>

где nnn – количество ссылок для соответствующего термина.

Комментарий:

Количество возвращаемых терминов может быть меньше Anumb, если достигнут конец словаря.

6.2 Получить список терминов словаря, начиная с заданного, и расформатировать записи, соответствующие первой ссылке каждого термина

function IC_nexttrmgroup(Adbn,Aterm: Pchar; Anumb: integer; Aformat: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:

Adbn,Aterm, Anumb – аналогичны одноименным параметрам функции IC_nexttrm;

Aformat – формат, который может задаваться пятью способами:

– строка непосредственного формата на языке форматирвания ИРБИС;

– имя файла формата, предваряемого символом @ (например @brief);

– символ @ - в этом случае производится ОПТИМИЗИРОВАННОЕ форматирование (т.е. имя формата определяется видом записи);

– символ * - в этом случае производится форматирование строго в соответствии со ссылкой (например для ссылки в виде 1.200.2.3 берется 2-е повторение 200-го поля);

– пустая строка. В этом случае возвращается только список терминов.

Во всех случаях перед форматированием выполняется следующая операция - в любом формате специальное *сочетание символов* вида *** (3 звездочки) заменяется на *значение метки поля*, взятое из 1-й ссылки для данного термина (например, для ссылки 1.200.1.1 формат вида v*** будет заменен на v200).

answer – буфер для возвращаемых данных;

abufsize – размер буфера.

Возвращаемые данные:

Код возврата – тот же, что и для функции IC_nexttrm;

answer –буфер в виде набора строк, каждая из которых имеет следующую структуру (в случае не пустого Aformat):

nnn#<ссылка>$30<термин>$30<результат расформатирования>

где:

nnn – количество ссылок для соответствующего термина;

<ссылка> - первая ссылка термина в виде mfn#tag#occ#pos#;

<результат расформатирования> - результат расформатирования, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130.

Если в качестве Aformat задано пустое значение, строки в буфере имеют структуру:
nnn#<термин>

Комментарий:
Количество возвращаемых терминов может быть меньше Anumb, если достигнут конец словаря.

6.3 Получить список терминов словаря, начиная с заданного, в обратном порядке

function IC_prevtrm(Adbn,Aterm: Pchar; Anumb: integer; answer: Pchar; abufsize: integer): integer;

Все аналогично функции IC_nexttrm.

6.4 Получить список терминов словаря, начиная с заданного, в обратном порядке и расформатировать записи, соответствующие первой ссылке каждого термина

function IC_prevtrmgroup(Adbn,Aterm: Pchar; Anumb: integer; Aformat: Pchar; answer: Pchar; abufsize: integer): integer;

Все аналогично функции IC_nexttrmgroup.

6.5 Получить список ссылок для заданного термина

function IC_posting(Adbn,Aterm: Pchar; Anumb,Afirst: integer; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Aterm – исходный термин;
Anumb – количество запрашиваемых ссылок; если задается 0 – будут выдаваться все ссылки термина, но не более, чем MAX_POSTINGS_IN_PACKET;
Afirst – порядковый номер первой из запрашиваемых ссылок; если задается 0 – то функция возвращает только общее количество ссылок заданного термина.
answer – буфер для возвращаемых данных;
abufsize – размер буфера.

Возвращаемые данные:
Код возврата – ZERO, если термин найден; общее число ссылок, если Afirst=0; или код ошибки;

answer – при коде возврата ZERO буфер содержит набор строк, каждая из которых является ссылкой вида:
mfn#tag#occ#pos#

Комментарий:
Для выделения отдельных составляющих ссылки следует пользоваться вспомогательной функцией IC_getposting.

6.6 Получить список первых ссылок для списка заданных терминов

function IC_postinggroup(Adbn,Aterms,answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Aterms – буфер со списком исходных терминов в виде набора строк;
answer – буфер для возвращаемых данных;
abufsize – размер буфера.

Возвращаемые данные:
Код возврата – определяется тем, найден ли первый термин из заданного списка: ZERO, если термин найден; код ошибки в противном случае;
answer – буфер, содержащий набор строк (по количеству исходных терминов), каждая из которых имеет структуру:
nnn#<ссылка>
где:
nnn – общее количество ссылок для соответствующего термина;
<ссылка> - первая ссылка термина.
Если соответствующий термин не найден в словаре, nnn=0, а <ссылка> - пустая строка.

6.7 Получить список ссылок для заданного термина и расформатировать записи им соответствующие

function IC_postingformat(Adbn,Aterm: Pchar; Anumb,Afirst: integer; Aformat: Pchar; answer1: Pchar; abufsize1: integer; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn,Aterm,Anumb,Afirst – аналогичны одноименным параметрам функции IC_posting;
Aformat – аналогичен одноименному параметру функции IC_nexttrmgroup;
answer1 – буфер для возвращаемых результатов расформатирования;
abufsize1 – размер буфера answer1;
answer – буфер для возвращаемого списка ссылок;

abufsize – размер буфера answer;

Возвращаемые данные:
Код возврата – определяется тем, найден ли исходный термин: ZERO, если найден; код ошибки в противном случае;
answer1 – содержит набор строк, каждая из которых имеет следующую структуру:
mfn#<результат расформатирования>
где:
mfn – MFN соответствующей записи;
<результат расформатирования> - результат расформатирования соответствующей записи, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130;
answer – содержит набор строк (соответствующих строкам в answer1), каждая из которых содержит ссылку в виде:
mfn#tag#occ#pos#

7. Функции поиска

7.1 Прямой (по словарю) поиск записей по заданному поисковому выражению

function IC_search(Adbn,Asexp: Pchar; Anumb,Afirst: integer; Aformat: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Asexp – поисковое выражение для прямого поиска на языке ИРБИС (см. Приложение 1);
Anumb – количество возвращаемых записей (из числа найденных); если задается 0 – возвращаются все найденные документы, но не более MAX_POSTINGS_IN_PACKET;
Afirst – номер первой возвращаемой записи из общего числа найденных; если задается 0 – возвращается только количество найденных записей;
Aformat – формат для расформатирования найденных записей, может быть задан следующими способами:

– строка непосредственного формата на языке форматирвания ИРБИС;

– имя файла формата, предваряемого символом @ (например @brief);

– символ @ - в этом случае производится ОПТИМИЗИРОВАННОЕ форматирование (т.е. имя формата определяется видом записи);

– пустая строка. В этом случае расформатирование записей не производится;

answer – буфер для возвращаемых данных;
abufsize – размер буфера answer.

Возвращаемые данные:
Код возврата – неотрицательное число, равное количеству найденных записей, или код ошибки;
answer –содержит набор строк, каждая из которых имеет структуру:
mfn#<результат расформатирования>
где:
mfn – MFN соответствующей записи;
<результат расформатирования> - результат расформатирования соответствующей записи, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130.

7.2 Последовательный поиск по результату прямого поиска и/или по заданному диапазону записей

function IC_searchscan(Adbn,Asexp: Pchar; Anumb,Afirst: integer; Aformat: Pchar; Amin,Amax: integer; Aseq: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn,Asexp,Anumb,Afirst,Aformat,answer,abufsize – аналогичны одноименным параметрам функции IC_search;
Amin,Amax – диапазон MFN, ограничивающий результат последовательного поиска; если в качестве Amin задан 0 – в качестве нижней границы принимается 1; если в качестве Amax задан 0 – в качестве верхней границы принимается максимальный mfn в базе данных; данные параметры учитываются только в том случае, когда задан последовательный поиск (т.е. когда Aseq не пустое выражение);
Aseq – поисковое выражение для последовательного поиска (представляет собой явный формат, который возвращает одно из двух значений: 0 – документ не соответствует критерию поиска, 1 – соответствует). Если задается выражение для прямого поиска (Asexp) – последовательный поиск ведется по его результату (с учетом Amin и Amax).

Возвращаемые данные:
Код возврата – неотрицательное число, равное количеству найденных записей, или код ошибки;
answer –содержит набор строк, каждая из которых имеет структуру:
mfn#<результат расформатирования>
где:
mfn – MFN соответствующей записи;
<результат расформатирования> - результат расформатирования соответствующей записи, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130.

8. Функции форматирования

Кроме функций, описанных в этом разделе, форматирование применяется в следующих функциях, описанных выше:
IC_readformat, IC_nexttrmgroup, IC_prevtrmgroup, IC_postingformat, IC_search, IC_searchscan.

8.1 Расформатирование записи, заданной по номеру (mfn)

function IC_sformat(Adbn: Pchar; Amfn: integer; Aformat: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;
Amfn – mfn исходной записи;
Aformat – формат для расформатирования исходной записи, может быть задан следующими способами:

– строка непосредственного формата на языке форматирвания ИРБИС;

– имя файла формата, предваряемого символом @ (например @brief);

– символ @ - в этом случае производится ОПТИМИЗИРОВАННОЕ форматирование (т.е. имя формата определяется видом записи);

answer – буфер для возвращаемых данных;
abufsize – размер буфера answer.

Возвращаемые данные:
Код возврата – код завершения форматирования или код ошибки;
answer – содержит результат расформатирования в виде набора строк.

8.2 Расформатирование записи в клиентском представлении

function IC_record_sformat(Adbn, Aformat,Arecord: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn, Aformat, answer, abufsize – аналогичны одноименным параметрам функции IC_sformat; (Кроме одного – для этой функции не работает конструкция Aformat=@);
Arecord – буфер, содержащий исходную запись в клиентском представлении (см. описание функции IC_read).

Возвращаемые данные:
Код возврата – код завершения форматирования или код ошибки;
answer – содержит результат расформатирования в виде набора строк.

8.3 Расформатирование группы записей

function IC_sformatgroup(Adbn,Amfnlist,Aformat: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn, Aformat, answer, abufsize – аналогичны одноименным параметрам функции IC_sformat;
Amfnlist – буфер, содержащий сведения об исходных записях; может быть двух видов:

1) набор из трех строк, следующего вида:
0
minmfn
maxmfn
где minmfn и maxmfn определяют диапазон mfn документов, подлежащих расформатированию;

2) набор строк следующего вида:
N
mfn1
mfn2
……
mfnN
где N – количество документов, подлежащих расформатированию.

Возвращаемые данные:
Код возврата – код завершения форматирования или код ошибки;
answer –содержит набор строк, каждая из которых имеет структуру:
mfn#<результат расформатирования>
где:
mfn – MFN соответствующей записи;
<результат расформатирования> - результат расформатирования соответствующей записи, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130.

9. Функции пакетной обработки

К данной группе относятся функции, реализующие пакетную обработку – вследствие чего они связаны с ДЛИТЕЛЬНЫМ выполнением (т.е. с длительным обращением к серверу ИРБИС64)

9.1 Формирование выходной табличной формы

function IC_print(Adbn,Atab,Ahead,Amod,Asexp: Pchar; Amin,Amax: integer; Aseq,Amfnlist: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn – имя базы данных;

Atab – имя табличной формы с предшествующим символом @ (например: @tabflw);
Ahead - заголовки над таблицей (до 3 строк). Реальные разделители строк $0D0A заменены на псевдоразделители $3130;
Amod – значение модельного поля, которое передается (только на период расформатирования) в каждую результирующую запись;
Asexp – поисковое выражение для прямого поиска на языке ИРБИС (см. Приложение 1);
Amin, Amax, Aseq – аналогичны одноименным параметрам функции IC_searchscan;
Amfnlist – список номеров (mfn) записей, организованный одним из трех следующих способов:

1) диапазон номеров – в виде трех строк следующей структуры:
   0
   minmfn
   maxmfn
2) список номеров – в виде набора строк:
   N
   mfn1
   mfn2
   …….
   mfnN
3) отрицательный список номеров («кроме указанных») – в виде набора строк:
   -N
   mfn1
   mfn2
   …….
   mfnN

Список результирующих документов формируется как результат пересечения трех списков:
- списка записей, найденных в результате прямого поиска (Asexp);
- списка записей, полученных в результате последовательного поиска (Amin,Amax,Aseq);
- списка записей, указанных с помощью Amfnlist.

answer – буфер для возвращаемых данных;
abufsize – размер буфера answer.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – содержит выходную табличную форму в формате RTF.

9.2 Формирование выходной формы в виде статистических распределений

function IC_stat(Adbn,Astat,Asexp: Pchar; Amin,Amax: integer; Aseq,Amfnlist: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn, Asexp, Amin, Amax, Aseq, Amfnlist, answer, abufsize – аналогичны одноименным параметрам функции IC_print;
Astat – список заданий на статистическую обработку, в виде набора строк, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130. Каждое задание представляет собой строку вида:

FMT,N1,N2,N3

где:

FMT – элемент данных, задаваемый в виде поле^подполе (например 200^a) или как непосредственный формат на языке форматирования ИРБИС;

N1 – анализируемая длина элемента данных;

N2 – максимальное количество значений элемента данных;

N3 – вид сортировки, который может принимать три значения:

0 – без сортировки;

1 – сортировка по убыванию;

2 – сортировка по возрастанию.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – содержит выходную форму в формате RTF.

9.3 Выполнение задания на глобальную корректировку

function IC_gbl(Adbn: Pchar; Aifupdate: integer; Agbl,Asexp: Pchar; Amin,Amax: integer; Aseq,Amfnlist: Pchar; answer: Pchar; abufsize: integer): integer;

Исходные данные:
Adbn, Asexp, Amin, Amax, Aseq, Amfnlist, answer, abufsize – аналогичны одноименным параметрам функции IC_print;
Aifupdate - параметр принимает два значения:

1 – актуализировать записи после корректировки;

0 – не актуализировать запись.

Agbl – задание на глобальную корректировку, которое может задаваться двумя способами:

1) имя файла задания с предшествующим символом @ (например: @glob);

2) в виде набора строк задания, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130. Описание структуры задания на глобальную корректировку см. в Общем описании системы.

Возвращаемые данные:

Код возврата – ZERO или код ошибки;
answer – содержит протокол выполнения глобальной корректировки..

10. Функции администратора

Выполнение функций данной группы возможно только в том случае, когда выполнена регистрация (IC_reg) с указанием в качестве типа клиента IRBIS_ADMINISTRATOR.

10.1 Перезапустить сервер ИРБИС64

function IC_adm_restartserver: integer;

Исходных данных нет.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.2 Получить список удаленных документов

function IC_adm_getdeletedlist(Adbn: Pchar; answer: Pchar; abufsize: integer):integer;

Исходные данные:
Adbn – имя базы данных;
answer – буфер для возвращаемых данных;
abufsize – размер выходного буфера.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – при успешном завершении содержит список удаленных записей в виде набора строк следующей структуры:

N#mfn

где:

N – принимает значения: 1 – запись удалена физически; 0 – запись удалена логически.

10.3 Получить общие сведения о базе данных: списки удаленных/заблокированных/неактуализированных, максимальный MFN и признак монопольной блокировки

function IC_adm_getalldeletedlists(Adbn: Pchar; answer: Pchar; abufsize: integer):integer; stdcall;

Исходные данные:

Adbn – имя базы данных;
answer – буфер для возвращаемых данных;
abufsize – размер выходного буфера.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – при успешном завершении содержит семь строк:

1 строка – общее количество удаленных, заблокированных и неактуализированных записей;

2 строка – список mfn записей удаленных логически в виде набора строк, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130;

3 строка – список mfn записей удаленных физически в виде набора строк, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130;

4 строка – список mfn неактуализированных записей в виде набора строк, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130;

5 строка – список mfn заблокированных записей в виде набора строк, в котором реальные разделители строк $0D0A заменены на псевдоразделители $3130;

6 строка – максимальный mfn+1;

7 строка – признак монопольной блокировки: 0 – нет блокировки; ERR_DBEWLOCK – монопольная блокировка

10.4 Опустошить базу данных

function IC_adm_dbempty(Adbn: Pchar):integer;

Исходные данные:
Adbn – имя базы данных.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.5 Удалить базу данных

function IC_adm_dbdelete(Adbn: Pchar):integer;

Исходные данные:
Adbn – имя базы данных.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.6 Создать новую базу данных электронного каталога

function IC_adm_newdb(Adbn,Adef: Pchar; AReader: integer):integer; stdcall;

Исходные данные:
Adbn – имя базы данных;
Adef – название базы данных на естественном языке;
AReader – признак доступности базы данных читателю: 1 – доступна; 0 – недоступна.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.7 Снять монопольную блокировку базы данных

function IC_adm_dbunlock(Adbn: Pchar):integer;

Исходные данные:
Adbn – имя базы данных.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.8 Снять блокировку заданных записей

function IC_adm_dbunlockmfn(Adbn: Pchar; Amfnlist: Pchar):integer;

Исходные данные:
Adbn – имя базы данных;
Amfnlist – список mfn записей в виде набора строк.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.9 Создать словарь базы данных заново

function IC_adm_dbstartcreatedictionry(Adbn: Pchar):integer;

Исходные данные:
Adbn – имя базы данных.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.10 Реорганизовать словарь базы данных

function IC_adm_dbstartreorgdictionry(Adbn: Pchar):integer;

Исходные данные:
Adbn – имя базы данных.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.11 Реорганизовать файл документов базы данных

function IC_adm_dbstartreorgmaster(Adbn: Pchar):integer;

Исходные данные:
Adbn – имя базы данных.

Возвращаемые данные:
Код возврата – ZERO или код ошибки.

10.12 Получить список зарегистрированных (текущих) клиентов

function IC_adm_getclientlist(answer: Pchar; abufsize: integer):integer;

Исходные данные:
answer – буфер для возвращаемых данных;
abufsize – размер выходного буфера.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – при успешном завершении содержит список текущих клиентов в виде набора строк (в ANSI-кодировке).

10.13 Получить список клиентов для доступа к серверу

function IC_adm_getclientslist(answer: Pchar; abufsize: integer):integer;

Исходные данные:
answer – буфер для возвращаемых данных;
abufsize – размер выходного буфера.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – при успешном завершении содержит список клиентов в виде набора строк (в ANSI-кодировке).

10.14 Получить список запущенных процессов на сервере

function IC_adm_getprocesslist(answer: Pchar; abufsize: integer):integer;

Исходные данные:
answer – буфер для возвращаемых данных;
abufsize – размер выходного буфера.

Возвращаемые данные:
Код возврата – ZERO или код ошибки;
answer – при успешном завершении содержит список запущенных процессов в виде набора строк (в ANSI-кодировке).

10.15 Обновить список клиентов для доступа к серверу

function IC_adm_setclientslist(AClientMnu: Pchar):integer;

Исходные данные:
AClientMnu – буфер с обновленным списком клиентов (в той же структуре, что возвращает функция IC_adm_getclientslist).

Возвращаемые данные:
Код возврата – ZERO или код ошибки;

11. Вспомогательные функции

Функции данной группы – кроме IC_nooperation – не связаны с обращением к серверу ИРБИС64.

11.1 Подтверждение регистрации

function IC_nooperation:integer;

Комментарий:
Функцию необходимо выполнять периодически с учетом того, что сервер автоматически разрегистрирует клиента, если от него не поступает никаких запросов в течение времени, которое определяется параметром CLIENT_TIME_LIVE в INI-файле сервера (irbis_server.ini).

11.2 Получить элемент исходной ссылки

function IC_getposting(APost: Pchar; AType: integer): integer;

Исходные данные:
APost – исходная ссылка;

AType – тип элемента ссылки; принимает следующие значения:: 0 – mfn; 1 – метка поля (точка входа - tag); 2 – номер повторения (occ); 3 – номер позиции (pos).

Возвращаемые данные:
Код возврата – соответствующий элемент ссылки.

11.3 Заменить реальные разделители строк $0D0A на псевдоразделители $3130

function IC_reset_delim(Aline: Pchar): Pchar;

Исходные данные:
Aline – исходный набор строк.

Возвращаемые данные
Aline – строка с псевдоразделителями.

11.4 Заменить псевдоразделители $3130 на реальные разделители строк $0D0A.

function IC_delim_reset(Aline: Pchar): Pchar;

Исходные данные:
Aline – строка с псевдоразделителями.

Возвращаемые данные
Aline – набор строк с реальными разделителями.

12. Функции для полнотекстовых баз данных

(пока нет)

Приложение 1. Поисковый язык ИРБИС

Поисковый язык ИРБИС предназначен для формулировки запросов, с помощью которых выполняется поиск в базах данных ИРБИС.
В ИРБИС выделяются два вида поиска:
Прямой поиск – быстрый поиск на основе инвертированного файла (т.е. на основе словарей);
Последовательный поиск – медленный поиск на основе последовательного перебора (просмотра) записей базы данных с возможностью выполнения таких операций, как БОЛЬШЕ/МЕНЬШЕ, НАЛИЧИЕ/ОТСУТСТВИЕ и др.

Язык прямого поиска.
Запрос для прямого поиска представляет собой алгебраическое (поисковое) выражение, в котором операндами являются термины словаря, а операторами

– логические операторы булевой алгебры. Для изменения порядка выполнения логических операторов в поисковом выражении могут применяться скобки.

Термин словаря включает в себя собственно термин словаря и префикс, если таковой используется для данного вида теминов.

В общем виде операнд поискового выражения можно представить следующим образом:

“<префикс><термин>$”/(tag1,tag2,…tagN)

где:

<префикс> - префикс, определяющий вид термина (вид словаря);

<термин> - собственно термин словаря;

$ - признак правого усечения термина; определяет совокупность терминов, имеющих начальную последовательность символов, совпадающую с указанным термином; может отсутствовать – в этом случае поиск идет по точному значению указанного термина.

“ – символ-ограничитель термина (двойные кавычки); должен использоваться обязательно, если термин включает в себя символы пробел, круглые скобки, решетка (#), а также символы, совпадающие с обозначениями логических операторов (см.. ниже);

/(tag1,tag2,…tagN) – конструкция квалификации термина; определяет метки поля, в которых должен находиться указанный термин, или точнее – вторую часть ссылки термина (см. Приложение 5 Общего описания системы ИРБИС); может отсутствовать – что означает отсутствие дополнительных требований в части меток полей.

В поисковом выражении могут использоваться следующие логические операторы:

- + - оператор логического ИЛИ; соединение двух операндов (терминов) логическим оператором ИЛИ обозначает требование поиска записей, в которых присутствует хотя бы один из терминов;
- * - оператор логического И; соединение двух терминов логическим оператором И обозначает требование поиска записей, в которых присутствуют оба термина;
- ^ - оператор логического НЕ; соединение двух терминов логическим оператором НЕ обозначает требование поиска записей, в которых присутствует первый термин и отсутствует второй; оператор НЕ не может быть одноместным (т.е. данному оператору, как и все м другим, должен ОБЯЗАТЕЛЬНО предшествовать термин);
- (G) – первый оператор контекстного И; соединение двух терминов таким оператором контекстного И обозначает требование поиска записей, в которых оба термина присутствуют в одном и том же поле (или точнее – когда у терминов совпадают вторые части ссылок);

- (F) – второй оператор контекстного И; соединение двух терминов таким оператором контекстного И обозначает требование поиска записей, в которых оба термина присутствуют в одном и том же повторении поля (или точнее – когда у терминов совпадают вторые и третьи части ссылок);
- . – (точка обрамленная пробелами) третий оператор контекстного И; соединение двух терминов таким оператором контекстного И обозначает требование поиска записей, в которых оба термина присутствуют в одном и том же повторении поля друг за другом (или точнее – когда у терминов совпадают вторые и третьи части ссылок, а третьи части ссылок отличаются на единицу);

Логические операторы имеют приоритеты, которые определяют порядок их выполнения (в пределах одного уровня скобок). Ниже операторы приведены в порядке убывания приоритета:

.
(F)
(G)
* и ^
+

Операторы одного приоритета выполняются слева направо (в пределах одного уровня скобок).

Для изменения порядка выполнения логических операторов в поисковом выражении могут применяться круглые скобки. Выражения в скобках могут объединяться только операторами + * ^.

Примеры запросов для прямого поиска:

(“A=Иванов$” +”A=Петров$”) * (“V=03” + “V=05”)

“K=трактор$” (F) “K=колесн$” + “K=бульдозер$” (F) “K=гусен$”

“K=очист$”/(200,922) * “K=вод$”/(200,922)

Язык последовательного поиска

Запрос (или поисковое выражение) для последовательного поиска представляет собой формат на языке форматирования ИРБИС (см. Приложение 4 Общего описания системы).

Собственно процесс последовательного поиска состоит в последовательном расформатировании исходных записей (т.е. тех записей, которые участвуют в поиске): если результатом расформатирования записи является строка ‘1’ (или точнее – строка, содержащая символ ‘1’), то соответствующая запись удовлетворяет поисковому запросу, в противном случае – не удовлетворяет.

Таким образом, в обобщенном виде запрос для последовательного поиска можно представить как следующий формат:

if <логическое выражение> then ‘1’ else ‘0’ fi

где **<логическое выражение>** - логическое выражение, в котором можно использовать все возможности языка форматирования ИРБИС.

Примечание: в АРМах Читатель и Каталогизатор ИРБИС в режиме ПОСЛЕДОВАТЕЛЬНЫЙ ПОИСК – СВОБОДНЫЙ ПОИСК в качестве запроса необходимо задавать именно **<логическое выражение>** (без объемлющей конструкции if … fi)

Последовательный поиск применяется только в тех случаях, когда нет возможности найти необходимые записи на основе прямого поиска (т.е. на основе словарей). Если некий последовательный поиск приходится применять регулярно, следует пересмотреть правила инверсии базы данных (<dbname>.fst) – с тем чтобы иметь возможность выполнять соответствующий поиск на основе словаря.