Ананас
Руководство хакера
Версия 1.0.0

Андрей Паскаль

27 января 2003 г.

Аннотация

В руководстве собрана информация о внутреннем устройстве программной системы, ее архитектуре. В нем вы найдете описание особенностей процесса компиляции, отладки исходного кода Ананаса и подготовки дистрибутива.

Данное руководство призвано координировать усилия участников проекта разработки программного обеспечения и документации Ананаса, оно позволяет лучше понять цели, направление и методы развития проекта.

Руководство предназначено для опытных программистов, желающих использовать Ананас в собственных разработках, самостоятельно вносить изменения в исходный код, выполнять локализацию или перенос Ананаса в другую операционную среду.

Содержание

1 Введение

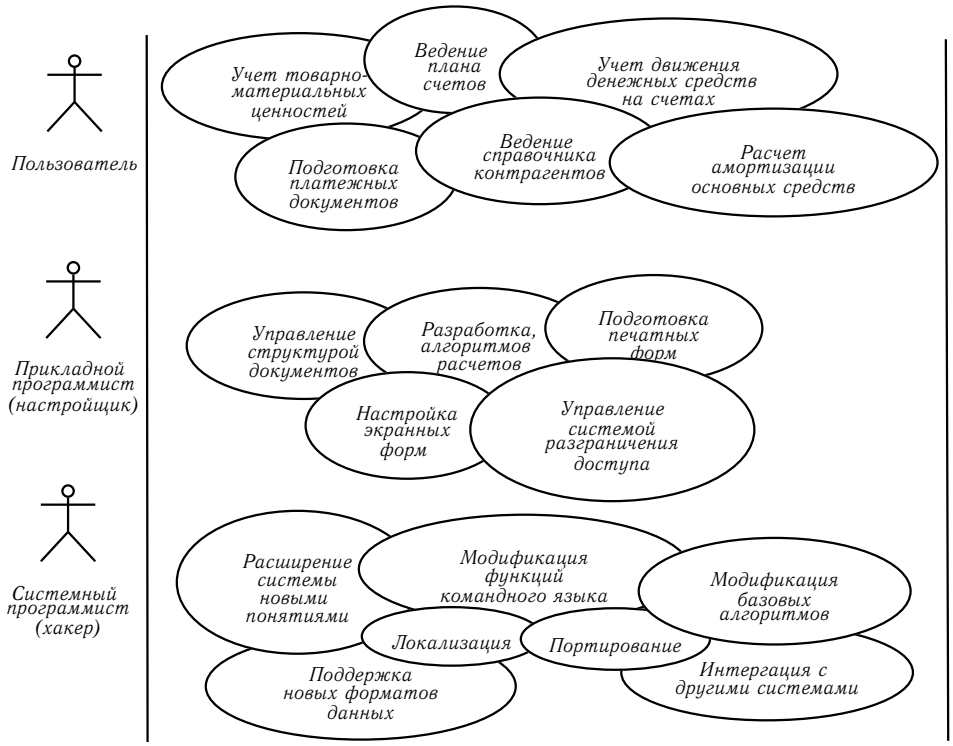
Ананас представляет собой специализированную программную платформу, предназначенную для разработки и сопровождения программно-информационных решений в области автоматизации финансово-хозяйственной деятельности предприятия.

В настоящий момент[1] разработан и поставляется только один модуль, созданный на платформе Ананас — Ананас.Склад, предназначенный для автоматизации ведения складского учета.

Поэтому большая часть материала, изложенного в настоящем руководстве касается именно этого модуля, точнее использования подсистем платформы Ананас в модуле Ананас.Склад.

2 Варианты использования

В настоящем разделе приведена модель вариантов использования, направляющая процесс разработки программного обеспечения Ананаса. Любой желающий осуществить самостоятельную доработку Ананаса, может оценить на сколько его доработки будут соответствовать модели вариантов использования. Варианты использования очерчивают не только набор функций, уже реализованных в проекте, но и могут также определять те функции, которые только еще планируется реализовать.



[1] По состоянию на май 2002 года

3 Архитектура

Ананас - сложная система и было бы не верно не приложить усилий к проектированию её архитектуры. Попросту говоря, проектирование архитектуры - это процесс разделения системы на части (подсистемы) и разработка протоколов (правил) взаимодействия подсистем между собой. С первого взгляда задача может показаться тривиальной, но на самом деле всегда есть возможность ошибиться. Для того, что бы этого не произошло нужно стараться как можно более ясно представлять себе цели архитектурного проектирования, выполняемого в данном конкретном случае и критерии выделения подсистем из единого целого.

Разбиение Ананаса на подсистемы не самоцель, есть веские причины поступать так. Эти причины, как ни удивительно, могут иметь не только техническую или технологическую природу. Я попробую вкратце высветить часть из них.

Необходимость снижения трудоемкости. В определенных обстоятельствах четкое разделение системы на подсистемы с хорошо определенными протоколами взаимодействия позволяет экономить труд за счет уменьшения объема работ. Рассмотрим пример с многоплатформенностью приложения. Если пред проектом стоит задача поддержки более одной платформы, не важно, выступает ли в роли платформы операционная система или сервер баз данных, то критерием “нарезки” подсистем будет критерий переносимости кода. То есть будет логичным выделение подсистем с непереносимым кодом, так что бы весь остальной код не требовалось модифицировать при переносе всего приложения на новую платформу. Это хорошо известный архитектурный прием, давно и успешно применяемый на практике. Что бы понять другой пример, необходимо рассматривать приложение или программную систему в ходе всего жизненного цикла. Даже достаточно простые программные продукты живут интересной и продолжительной жизнью после своего первоначального выпуска в свет. Появляются новые версии, добавляются новые функции, перестают использоваться старые. То есть программам все время приходится перестраиваться, адаптируясь к меняющемуся программному окружению их существования, к требованиям пользователя, новым стандартам на представление, хранение и обмен данными. Ценой, которую приходится платить разработчику за адаптацию часто является переписывание кода системы. Если система изначально была грамотно спроектирована, разбита на функциональные подсистемы, то переписывание может ограничится рамками одной двух подсистем. При этом обязательно будут подсистемы, чей код на протяжении продолжительного времени будет оставаться неизменным. Вот пример из "жизни"Ананаса. При переходе от версии 0.1.x к версии 0.3.x была переписана большая часть кода, но подсистема “Отчеты” смогла пережить “революцию” и перекочевала в новую версию без каких-либо изменений.

Разделение работы между разработчиками. Вполне естественно работа может быть разделена между несколькими программистами в том случае, если сама система четко разделена на функциональные модули. Даже если в самом начале рождения проекта разработкой занимается одни единственный программист (он же по совместительству и автор идеи), все равно эта причина выделения подсистем остается в силе. Во-первых, сила модели открытых исходников заключается в том, что проект разрабатывается открыто и приветствуется участие в нем всех желающих, способных внести в проект хоть что-то ценное. Поэтому, взявший на себя смелость и мужество

по запуску открытого проекта, обязательно должен постараться привлечь как можно больше программистов и позаботиться о том как грамотно разделить работу между теми, кто придет в проект. Таким образом получается, что грамотная архитектура проекта становится дополнительной предпосылкой для принятия проекта другими программистами и их участия в его развитии. В любом случае координировать работу по такому проекту будет гораздо проще.

Использование готовых подсистем со стороны. Проекты с открытыми исходниками пользуются такой привелегией как наличие большого количества свободно доступных готовых подсистем, библиотек. Такие заимствованные подсистемы могут составлять существенную часть всего приложения и быть естественными частями архитектуры системы. Так например, вполне разумно использовать уже готовый командный язык, выбрав наиболее подходящий из всего разнообразия имеющихся, а не разрабатывать новый.

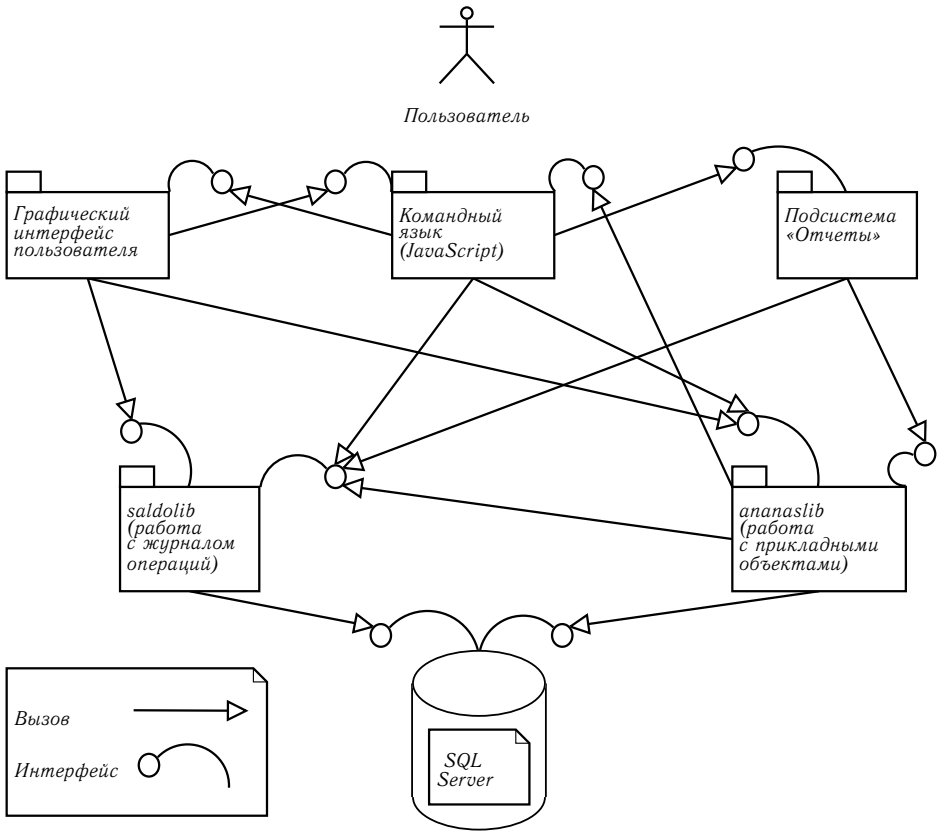
На схеме ниже изображена архитектура системы, показаны интерфейсы модулей и вызовы. В принципе, нет ограничений или специальных правил, определяющих какой модуль может вызывать функции того или иного модуля. Модули рассматриваются как равнозначные. Пожалуй единственным правилом в текущей версии Ананаса является следующее. Только модули saldolib и ananaslib могут на прямую обращаться к SQL серверу. Все остальные модули, которым нужен доступ к данным, хранящимся в базе данных SQL сервера, должны использовать программный интерфейс, предоставляемый модулями saldolib и ananslib. Это сделано для того, чтобы локализовать код, зависимый от SQL сервера, с тем что бы облегчить в последствии переход к другим SQL серверам.

В ходе развития проекта вполне естественно появление новых модулей, которым будет недостаточно программного интерфейса saldolib и ananaslib. В такой ситуации предполагается доработка программного интерфейса, добавление в него новых функций.

По сути, ananaslib и saldolib представляют собой ядро Ананаса. В настоящий момент saldolib рассматривается как наиболее сформировавшаяся, универсальная и стабильная[2] часть ядра. Предполагается, что при дальнейшем развитии Ананаса в краткосрочной перспективе изменения, которые будет претерпевать ядро, главным образом коснуться ananaslib, так как эта библиотека ближе к прикладной части Ананаса.

В приведенной схеме модуль “Командный язык” является планируемым. То есть он отсутствует в Ананасе версии 0.3.x и запланирован для введения в следующих версиях Ананаса.

[2]Стабильная не в смысле отсутствия ошибок, а в том плане, что она реализует концептуально законченную часть ядра Ананаса. Имеется в виду, что множество реализуемых функций является достаточным для выполнения всех/любых операций с остатками



4 Цикл “компилирование-установка-отладка”

Значительную часть времени разработки Ананаса приходится тратить на отладку вновь написанного кода. В этом разделе кратко описана последовательность действий, традиционно выполняемых кодировщиком в ходе работ над проектом.

4.1 Подготовка “рабочего места”

Первое, что нужно для работы с кодом Ананаса, естественно сам код :)

Поэтому, в каком-нибудь рабочем каталоге[3] распакуйте архив с исходниками командой

tar -zxvf ananas-0.3.x.tgz

Если вы имеете счастливую[4] возможность работать с CVS, задайте правильное значение CVSROOT и дайте команду

cvs co ananas

В результате, вы получите дерево исходников Ананаса. Следующая команда

cd ananas; make

[3] я обычно работаю в $HOME/devel

[4] CVS Ананаса все еще не находится в постоянном доступе в Интернет

выполнит компиляцию исходного кода и создаст в каталоге ananas/bin исполняемый файл[5] ananas.

Теперь можно выполнить установку Ананаса в вашу систему. Не смотря на то, что установка будет происходить без использования программы rpm, вас не должна беспокоить проблема деинсталляции. Вы сможете деинсталлировать Ананас ровно одной командой. Установка Ананаса из откомпилированного дерева исходников отличается от установки дистрибутива Ананаса при помощи команды rpm тем, что в первом случае не создается база данных на SQL сервере. Базу данных, а точнее две базы данных, нужно будет создать отдельно. Эта процедура описана ниже.

4.2 Подготовка баз данных mysql

В каталоге ananas/src/sql находятся файлы с sql командами, создающими необходимые структуры данных на сервере MySQL. MySQL может быть установлен и запущен как локально[6], так и удаленно[7]. Файлы, имя которых содержит префикс “create_”, отвечают за создание таблиц в базе данных. Саму базу данных необходимо создать командой

mysqladmin -u root[8] create ananas_db

Для тестирования Ананаса удобно иметь демонстрационную базу данных, уже наполненную тестовыми данными. Создадим её командой

mysqladmin -u root create ananas_demo

для наполнения демонстрационной базы тестовыми данными следует выполнить команду[9]

mysql -u root ananas_demo < ananas/src/sql/demodb.sql

А для создания пустых таблиц базы данных Ананаса, которая обычно называется Рабочей, следует дать команду

cat ananas/src/sql/create_*|mysql -u root ananas_db

Важно иметь ввиду, что удаление баз данных, при необходимости, нужно будет так же выполнять вручную командами

mysqladmin -u root drop ananas_db

mysqladmin -u root drop ananas_demo

так как удаление Ананаса командой make ни как не затрагивает базы данных на MySQL сервере.

4.3 Первое компилирование

Компилирование Ананаса выполняется командой

cd ananas; make

Необходимо заметить, что выполнение команды make в каталоге самого верхнего уровня дерева исходников Ананаса приводит к компилированию заимствованной

[5] описание каталогов дерева исходников Ананаса смотрите в следующих разделах настоящего руководства

[6] На той же машине, что и Ананас

[7] На любой другой машине, доступной по сети

[8] Предполагается, что пользователь root имеет пустой пароль доступа к MySQL серверу, то есть его пароль не менялся после установки MySQL сервера

[9] В файле demodb.sql находятся также команды создания необходимых таблиц базы данных

библиотеки Xbase, отвечающей за работу с DBF[10] файлами. Так как в ходе работы над Ананасом не предполагается доработка, этой библиотеки, то будет разумным не запускать команду make, каждый раз приводящую к пересборке Xbase, в самом верхнем каталоге дерева исходников Ананаса[11]. Вместо этого следует выполнять команду make в каталоге ananas/src или в одном из его подкаталогов, где также присутствует файл Makefile.

4.4 Установка-удаление командой make

После успешной сборки Ананаса, пользователь с правами root может установить его командой

cd ananas;make install

Если в вашей системе уже был установлен Ананас, либо аналогичным способом, либо из rpm пакета, то предварительно необходимо провести деинсталляцию, ранее установленной версии. Иначе при установке возникнут ошибки.

Если установка прошла успешно и вы подготовили базу данных, следуя инструкциям предыдущих разделов, можете запустить Ананас и проверить как он работает. Запуск осуществляется командой

ananas.sklad

Удаление установленного с помощью приведенной выше команды Ананаса выполняется суперпользователем командой

cd ananas;make uninstall

4.5 Цикл отладки

Порядок отладки зависит от того, с какой частью системы вы работаете. Я покажу как можно работать, внося исправления в Си код Ананаса и в формы отчетов, хранящиеся в TEX формате.

Рассмотрим случай, когда идет работа по исправлению Си кода, например, в каталоге ananas/src/kui. Как нетрудно догадаться, код находящийся файлах этого каталога, отвечает за работу графического интерфейса пользователя. И пусть вас не смущает, что название каталога начинается с буквы k, которая подразумевает KDE. На самом деле, для создания пользовательского интерфейса намеренно не используются элементы KDE интерфейса. Выбор сделан в пользу использования исключительно QT библиотеки, обеспечивающей переносимость между разными платформами, такими как Windows, Mac и Linux.

Итак, допустим, вы внесли свои изменения в код и хотите посмотреть как поведет себя Ананас с вашими изменениями. (Предполагается, что немодифицированный, оригинальный Ананас уже установлен у вас в соответствии с инструкциями предыдущих разделов.)

[10] Раньше в Ананасе версии 0.1.x dbf файлы использовались как основной способ хранения данных. В настоящее время dbf файлы используются Ананасом только для передачи данных в систему генерирования отчетов

[11] Этот исторически сложившийся подход мы надеемся исправить в ближайшее время, выделив Xbase в отдельный rpm пакет

Находясь в этом же (ananas/src/kui) каталоге, выполните команду

make

если сборка прошла успешно, то все, что осталось сделать перед запуском модифицированного Ананаса - скопировать суперпользователем “свежеиспеченный бинарник” командой

cp ananas/bin/ananas /usr/bin/

После этого, запускайте Ананас как обычно, командой

ananas.sklad

и наслаждайтесь результатами своей работы.

Если вы вносите исправления в код ядра Ананаса, расположенный в каталоге ananas/src/lib, то для получения “свежеиспеченного бинаркина”, находясь в каталоге ananas/src/lib, необходимо выполнить две команды make.

make; cd ../kui;make

В случае с внесением изменений в файлы с описанием отчетов можно поступить следующим образом. Исправлять файлы в месте их установки. Для Ананаса версии 0.3.x - это каталог /usr/share/ananas/rep

Ананаса не нужно перезапускать каждый раз после исправления того или иного rep файла.

И самое главное, не забудьте скопировать отлаженный rep файл в дерево исходников, если вы хотите, что бы он там остался.

4.6 Подготовка диалогов в QT Designer и локализация

Необходимость в этом разделе возникла потому, что Qt Designer хотя и поддерживает локализацию, делает это с небольшой ошибкой, исключающей возможность применения кириллических надписей во время проектирования диалога. Точнее надписи то делать можно и они даже корректно отображаются в режиме preview. Но вот зато cpp код, сгенерированный из ui файла при помощи uic, будет некорректно выводить кириллические надписи. Эта ошибка существует по крайней мере до версии 3.0.0 библиотеки qt. Что бы обойти эти проблемы, в проекте Ананас принят следующий подход. При проектировании интерфейса средствами Qt Designer все надписи делаются латиницей. Затем с помощью Qt Linguist выполняется локализация (перевод на русский язык). Файл локализации устанавливается вместе с дистрибутивом и используется Ананасом.

5 Подготовка дистрибутива

Если вы внесли существенные исправления в код Ананаса, то логично создать дистрибутив, оформив его в виде rpm пакета. В такой форме гораздо проще представить ваши достижения, тому, кто захочет на них самостоятельно посмотреть.

5.1 Дерево каталогов исходного кода

Исходный код Ананаса разбит на множество файлов, которые расположены в разных каталогах на разных уровнях дерева каталогов Ананаса.

После распаковки файла архива ananas-0.3.1.tgz с исходным кодом Ананаса, распространяемым через официальный сайт проекта Ананас, вы получите каталог ananas-0.3.1 . В этом каталоге находятся файлы changelog.ru, COPYING, COPYING.RU, Makefile, README.RU.in, todo, version, а также каталоги bin, data, db, doc, enitl, examples, origin, rep, rpm, skel, src, xbase-1.8.1 .

Файл changelog.ru, как нетрудно догадаться по его названию, содержит описание изменений, вносимых в проект в ходе создания новых версий. В нем приведен список выпущенных версий с пояснениями по каждой версии.

Файлы COPYING и COPYING.RU содержат текст лицензионного соглашения на использование Ананаса на английском и русском языке соответственно.

Makefile — это файл, используемый командой make для компилирования (сборки) исходного кода.

README.RU.in содержит краткое описание процесса установки дистрибутива Ананаса. В нем перечислены требования к программному обеспечению, необходимому для установки и эксплуатации Ананаса.

Краткий перечень планируемых в проекте изменений и доработок расположен в файле todo. Этот перечень нововведений периодически пересматривается и отражает взгляд разработчиков на перспективы развития проекта на момент выхода очередного релиза. При этом нет ни

какой гарантии, что при выходе следующего релиза эти взгляды не претерпят изменений.

О назначении файла version подробно рассказывается в следующем разделе настоящего руководства.

Теперь рассмотрим назначение каталогов.

5.2 Задание версии Ананас

Перед сборкой дистрибутива версию Ананаса нужно прописать в файле ananas/version. Во всех других файлах, где требуется указать номер версии Ананаса, следует писать ANANASVER. Этот макрос будет заменен на номер версии в результате работы скрипта, находящегося в Makefile.

5.3 Подготовка rpm файла дистрибутива

Для создания rpm файла дистрибутива из CVS репозитария в Red Hat Linux необходимо выполнить следующую последовательность действий.

1. Зарегистрироваться в системе с правами супер пользователя, командой su

2. Перейти в каталог /usr/src/redhat/SOURCE[12]

3. Убедиться, что переменная среды окружения CVSROOT указывает на CVS репозитарий, в котором находится модуль ananas. Для этого достаточно просмотреть результат работы команды set|grep CVSROOT

[12] Если в вашей системе нет каталога /usr/src/redhat, значит вы не установили пакет rpm-build.

4. Получить исходный код проекта, выполнив команду экспорта.

cvs export -r release-0-3-1 ananas[13]

Эта команда создаст в текущем каталоге подкаталог ananas и запишет туда исходный код Ананаса версии 0.3.1

5. Создать копию каталога ananas

cp -Rf ananas ananas-0.3.1

6. Создать архив дистрибутива

tar -zcvf ananas-0.3.1.tgz ananas-0.3.1

7. Подготовить файлы ananas.spec и Makefile

cd ananas; make rpm/Makefile rpm/ananas.spec; cd ..

8. Переместить их в требуемые места

mv -f ananas/rpm/Makefile ananas mv -f ananas/rpm/ananas.spec ../SPECS

9. Подготовить патч командой

diff -c ananas-0.3.1 ananas > ananas-0.3.1.patch

10. На этом процесс подготовки заканчивается и все что останется сделать для получения дистрибутива это перейти в каталог /usr/src/redhat/SPECS и набрать команду

rpm -ba ananas.spec

Скорость компилирования Ананаса очень сильно зависит от производительности вашей машины. На моём Celeron-366 со 192Мб ОЗУ дистрибутив собирается не меньше десяти минут. Так что самое время идти пить чай, или перекусить что-нибудь на кухне.

Если все прошло успешно, то в каталоге /usr/src/redhat/RPMS/i386/ вы найдете заветный файл.

6 Добавление новых баз данных

Для того чтобы при запуске Ананаса пользователь в списке доступных баз видел не две, а скажем три, вам потребуется создать новую базу данных, создать каталог с конфигурационным файлом и подправить содержимое другого конфигурационного файла.

Создание БД выполняется следующими двумя командами

```
mysqladmin -u root create mydb3
cat ananas/src/sql/create_*|mysql -u root mydb3
```

В результате вы получите пустую БД, готовую к работе. Теперь нужно дать знать Ананасу, что появилась еще одна база данных, с которой он может работать. Для этого загружаем в ваш любимый текстовый редактор файл /etc/ananasrc[14] и добавляем в него с новой строки следующий текст

```
db_title3=Моя новая база данный номер 3
```

[13] Нужно иметь ввиду, что вам следует указывать в этой команде не release-0-3-1, а имя тэга соответствующего той версии Ананаса, дистрибутив которой вы хотите получить

[14] или файл ~/.ananasrc, если таковой имеется

db_path3=/usr/share/ananas/db3

Теперь нужно создать каталог db3

mkdir /usr/share/ananas/db3

и создать в новом каталоге файл .myrc, содержащий параметры доступа к sql серверу.

db_host=localhost
db_name=mydb3
db_login=root
db_password=

Все, теперь можно запустить Ананас и убедиться, что все работает.

7 Ананас и учет движения средств по счетам

Несмотря на то, что Ананас в настоящее время используется как программа складского учета, в последующих версиях функциональность будет расширена. Ананас предоставит пользователям возможность ведения плана счетов, проводок, формирования баланса, оборотно-сальдовой ведомости и других бухгалтерских документов, выполнения анализа операций по счетам.

7.1 План счетов. Проводки

Уже начиная с версии 0.3.1 для учета движения товара в Ананасе используются проводки и журнал операций. Складские операции по поступлению и реализации товаров отражаются следующими проводками.

Приход - Дт.41.2-Кт.60

Расход - двумя проводками

Дт.46-Кт.41.2

Дт.46-Кт.42

Вторая проводка необходима для отдельного учета торговой наценки на счете 42.

Ниже поясняется назначение счетов. В скобках указано название счета в соответствии с бухгалтерским планом счетов.

60 - сумма закупленного товара (Расчеты с поставщиками и подрядчиками)

41.2 - сумма остатка товаров на складе по закупочным ценам (Товары в розничной торговле)

46 - сумма отпущенного (реализованного) товара с учетом наценки (Реализация продукции (работ, услуг)[15])

42 - сумма наценки по отпущенным (реализованным) товарам (Торговая наценка)

[15]В новом плане счетов, вступившем в силу с 1-го января 2002 года, вместо счета 46 используется счет 90

Все эти данные я привожу именно в руководстве хакера, а не в руководстве пользователя и не в руководстве программиста только потому, что и журнал операций и проводки скрыты внутри системы и не доступны для пользователя и программиста-настройщика. Эта ситуация будет исправлена с выходом версии Ананаса, более приспособленной для ведения бухгалтерского учета. Пока этого не произошло для доступа к проводкам, формируемым Ананасом, и информации об остатках необходимо использовать API библиотеки saldolib или на прямую обращаться к таблице oplog соответствующей базы данных Ананаса, хранимой на SQL сервере.

7.2 Расчет и хранение остатков в saldolib

Ананас выполняет автоматический расчет остатков по счетам в момент добавления (удаления или модификации) проводки в журнал операций. Для вычисления остатка по счету используются только данные журнала операций, хранящиеся в таблице oplog соответствующей базы данных SQL сервера. Информация по остаткам на счетах хранится в таблице saldo той же базы данных того же SQL сервера.

Все операции с SQL таблицами saldo и oplog выполняются только библиотекой saldolib. И если какому-либо модулю Ананаса необходим доступ к журналу операций или информации об остатках, то он обращается к функциям библиотеки saldolib.

Для вычисления остатков saldolib использует следующий

“Алготитм добавления проводок в журнал операций и расчета остатков на их основе”

begin transaction

```
/*
 Первым делом запишем проводку в журнал операций
 */

insert into oplog (tdate,debit,kredit,sum) values(“01.01.2001”,“61.1”,“51.1”,100);

/*
 Дальше пошли операции с остатками
 */

select vsego as bufX from saldo where tdate =“01.01.2001” and acn=“61.1”;
select vsego as bufY from saldo where tdate=“01.01.2001” and acn=“51.1”;
delete from saldo where tdate=“01.01.2001” and (acn=“61.1” or acn=“51.1”);

/*Вычисление остатка по счету 61.1 ниже*/
select sum(vsego) as S1 from oplog where tdate =“01.01.2001” and debit=“61.1”;
select sum(vsego) as S2 from oplog where tdate =“01.01.2001” and kredit=“61.1”;
select max(tdate) as Z from saldo where acn=“61.1” and tdate < “01.01.2001”;
select vsego as S3 from saldo where acn=“61.1” and tdate=Z;

/*
 S1 - сумма по дебету данного счета на данный день
 S2 - сумма по кредиту данного счета
```

```
  S3 - остаток по данному счету на предыдущий день
  Z - дата предыдущего дня
  X:=S1-S2+S3
  Y вычисляется аналогично X, заменой во всех четырех SQL командах
  номера счета с 61.1 на 51.1
*/

insert into saldo (tdate, acn, vsego) values ("01.01.2001", "61.1", X);

/*
  Теперь нужно обновить проводки с более поздней датой чем 01.01.2001,
  зависящие от измененных только что данных, если таковые проводки имеются
  */

update saldo set vsego=vsego+(X-bufX) where acn="61.1"and tdate>"01.01.2001";

/*
  Теперь повторяем две последние операции, но уже для счета 51.1
  */

insert into saldo (tdate, acn, vsego) values ("01.01.2001","51.1",Y);
update saldo set vsego=vsego+(Y-bufY) where acn="51.1"and tdate>"01.01.2001";
end transaction
```

Необходимо заметить, что приведенный алгоритм является схематичным и показывает расчет остатков для синтетического учета. Однако, он позволяет понять как организовано хранение остатков в Ананасе. Для детального знакомства с порядком ведения журнала операций и расчета остатков при аналитическом учете следует ознакомится с содержимым файлов ananas_saldo.c и ananas_oplog.c из каталога src. Содержащиеся там комментарии должны облегчить изучение кода.

8 Использование Ананаса в Си программах

При проектировании структуры Ананаса особое внимание уделялось и уделяется возможности интегрирования кода Ананаса с другими программными системами.

С точки зрения программиста Ананас можно рассматривать как набор Си библиотек, заключающих в себе всю функциональность Ананаса. Благодаря этому любая функция, реализуемая Ананасом, доступна через подключение и вызовы Си функций соответствующих библиотек.

Каждая библиотека имеет свой заголовочный .h файл, в котором определены все доступные функции библиотеки. Совокупность Си функций, определенных во всех заголовочных файлах представляет из себя AnanasAPI - программный интерфейс Ананаса[16].

[16] Детальное описание AnanasAPI см. в разделе ... настоящего руководства.

В настоящий момент[17] AnanasAPI представлен интерфейсами трех библиотек:
- ananassaldolib;
- ananaslib;
- ananaskuilib.

Каждая из этих библиотек объединяет логически связанный набор функций. Кроме этого библиотека ananaslib включпет в себя библиотеку ananassaldolib, а библиотека ananaskuilib включает в себя ananaslib и соответственно ananassaldolib.

То есть для того, что бы использовать в своей программе функции всех трех библиотек, достаточно в параметрах линковщика указать -lananaskuilib.

8.1 Библиотека ananassaldolib

ananassaldolib обеспечивает функции, необходимые для работы с журналом операций (проводок) и с остатками. Проводки журнала операций и информация об остатках хранятся в SQL таблицах.[18] Поэтому прежде чем использовать библиотеку ananassaldolib вы должны убедится, что на SQL сервере присутствуют требуемые таблицы. Иначе вызываемые вашей программой функции будут возвращать ошибки.

Если вы ранее успешно установили **Ананас** в своей системе, то смело можете использовать ananassaldolib.

Основными функциями ananassaldolib являются добавление, удаление, чтение и изменение проводок журнала операций и функции получения информации об остатках. Информация об остатках доступна только для чтения.

При работе с ananassaldolib важно понимать, что изменения, вносимые в журнал операций, автоматически отражаются на информации об остатках. То есть приводят к ее изменению.

8.2 Журнал операций

С точки зрения программиста, использующего AnanasAPI для работы с журналом операций, он представляет из себя таблицу из десяти столбцов, в каждой строке которой хранится информация ровно об одной проводке.

Концепция журнала операций и проводок, хранящихся в нем, позаимствована у разработчиков 1С:Бухгалтерии. Поэтому, тот кто знаком с этой системой без труда поймет организацию журнала операций **Ананаса**.

Журнал операций используется в ananas.sklad - складском модуле **Ананаса**, хотя основное его назначение - это ведение учета движения денежных средств по бухгалтерским счетам в модуле бухгалтерского учета ananas.balance.

В модуле ananas.sklad заполнение журнала операций происходит при сохранении пользователем программы одного из двух типов документов: проходной накладной или расходной накладной.

---

[17] **Ананас** версии 0.3.x

[18] Библиотека ananassaldolib работает с двумя таблицами базы данных **Ананаса**, хранящимися на SQL сервере. Это таблица oplog, которая используется для хранения проводок журнала операций, и таблица saldo, которая используется для хранения информации об остатках. Описание структуры каждой таблицы смотрите в разделе ... настоящего руководства.

Каждая проводка (запись) журнала операций содержит ссылку на породивший ее документ. Если пользователь удаляет тот или иной документ, то из журнала операций автоматически удаляются проводки (записи) соответствующие этому документу.
Как при добавлении новых проводок в журнал операций, так и при удалении проводок из него происходит автоматическая корректировка остатков на счетах, задействованных в этих проводках.
Каждая проводка журнала операций содержит следующую информацию:
- дебетуемый счет - символьное значение, идентифицирующее бухгалтерский счет проводки, в дебет которого заносится сумма проводки.
- кредитуемый счет - символьное значение, идентифицирующее бухгалтерский счет проводки, в кредит которого заносится сумма проводки.
- сумма проводки - число с двумя знаками после запятой, указывающее сумму, переносимую с одного счета на другой
- количество - число с двумя знаками после запятой, указывающее количество товара в денежном выражении.
- субконто по дебету -
- субконто по кредиту -
- примечание -
- дата -

В полях "Субконто по дебету"и "Субконто по кредиту"могут хранится либо ссылка на субконто (идентификатор субконто), либо комбинированное значение.
То есть библиотека ananassaldolib ни как не ограничивает системного программиста в выборе модели организации данных и связей между элементами данных. Расчет остатков, осуществляемый автоматически библиотекой ananassaldolib, ведется для каждого уникального ключа "дата, счет, субконто". Каждая запись журнала операций дает два таких ключа:
1. дата, счет по дебету, дебетовое субконто
2. дата, счет по кредиту, кредитовое субконто
Поэтому каждой записи в журнале операций соответствует две записи в таблице остатков. То есть добавление одной операции (проводки) приводит к добавлению двух записей в таблицу остатков[19]

8.3 Использование ananassaldolib модулем ananas.sklad

В предыдущем разделе дано описание организации библиотеки ananassaldolib и порядка работы с ней без относительно к какой либо программе, ее использующей. В настоящем разделе рассказывается о том как эта библиотека используется модулем ananas.sklad.
Специфика использования библиотеки модулем ananas.sklad связана с:
1. необходимостью вести учет одного и того же наименования товара, поступившего

[19]Перед добавлением новых записей в таблицу остатков происходит удаление из нее прежних записей с соответствующим ключем. Детальное описание алгоритма формирования таблицы остатков смотрите в следующем разделе

по разной цене, как двух разных товаров.
2. моделью организации данных принятой в модуле "Склад", когда в журнал операций записываются не ссылки на субконто (идентификаторы субконто), а их наименования. А справочник субконто используется исключительно для облегчения ввода большого количества данных в документы.
Приведенная в этом разделе информация важна только в том случае, если вы планируете осуществить доработку модуля "Склад"своими силами так, что бы сохранилась совместимость по структуре данных с оригинальной версией модуля "Склад". Если же вы не стремитесь к сохранению совместимости, то приведенную в этом разделе информацию можно рассматривать как пример организации работы с библиотекой ananassaldolib.
Итак, основной и единственной особенностью использования библиотеки ananassaldolib модулем "Склад"является структура строки идентифицирующей субконто, которая передается в параметрах соответствующих функций библиотеки.
Для идентификации субконто используется строка следующего вида:
«a=1>Группа товаров<a=2>Наименование товара<a=3>Закупочная цена"
Пример:
«a=1>ППа><a=2>Пиво Толстяк<a=3>15.00"
Еще раз необходимо подчеркнуть, что библиотека ananassaldolib не имеет ни малейшего представления о структуре строки, идентифицирующей субконто. О формировании и разборе строки должна позаботится программа, использующая библиотеку. В нашем случае эту работу выполняет модуль "Склад"[20].
Приведенный выше формат идентификатора субконто показывает, как идентифицируются товары в модели данных модуля "Склад".
Но в качестве субконто могут выступать не только товары, но также и контрагенты (поставщики, покупатели)

A Структура базы данных

В приложении приводится описание структуры хранения информации в базе данных программы ananas.sklad версии 0.3.x. В качестве СУБД использован реляционный сервер баз данных с поддержкой транзакций MySQL, распространяемый на условиях лицензии GPL. SQL запросы, создающие таблицы, описанные в настоящем документе, находятся в каталоге src/sql дерева исходного кода проекта Ананас. Все описанные таблицы хранятся в одной базе данных. Возможно использовать для ведение учета более одной базы данных. В этом случае, пользователем выбирается требуемая база данных в момент запуска программы.

[20] Да, я понимаю, что такая организация данных не является правильной с точки зрения теории реляционных баз данных, так как не удовлетворяет требованиям даже первой нормальной формы представления данных. Но такова жизнь. И стремление к универсальности заставляет порой выходить за рамки чистой теории

A.1 Таблицы хранения расходных накладных

В каждой записи таблицы rn хранятся атрибуты одной расходной накладной. Все кроме информации о позициях накладной. В таблице rns хранятся позиции накладной. Между таблицами существует связь типа “один-ко-многим” rn.id<=>>rns.rn_id

Пояснения к полям таблицы “rn”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	уникальный идентификатор расходной накладной (записи в таблицы) - первичный ключ. Значения генерируются и заносятся программой. Пользователь не имеет доступа к этому полю.
2.	doctype	char	5	Символьное поле для хранения типа документа (не используется в настоящее время).
3.	docnum	int	-	номер накладной, задаваемый пользователем. Может существовать несколько накладных с одинаковыми номерами. Символьное поле длиной 9 символов.
4.	docdate	date	8	Дата расходной накладной
5.	client	char	80	Наименование покупателя. Это не ссылка, а именно наименование.
6.	summa	float	-	Сумма по накладной
7.	prim	char	50	Используется при начислении налога с продаж. Если имеет значение “НП 5%”, то по этой накладной необходимо начислять налог с продаж. Иначе поле пустое. Также используется при печати отчетов.
8.	nakrutka	float	-	Наценка по накладной
9.	reason	char	160	Основание. Значение данного поля без изменений пропечатывается в одноименном поле бумажной накладной

пояснения к полям таблицы “rns”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи таблицы. Первичный ключ.
2.	rn_id	int	-	Идентификатор расходной накладной - внешняя ссылка на поле id таблицы rn. Значения генерируются и заносятся программой. Пользователь не имеет доступа к этому полю.
3.	tip	char	5	Наименование группы товаров, к которой принадлежит данный товар
4.	name	char	50	Наименование товара. Это не ссылка, а именно наименование.
5.	kol	float	-	Количество отпускаемого товара.
6.	price	float	-	Отпускная цена товара.
7.	summa	float	-	Сумма по товару.
8.	nakrutka	float	-	Наценка по товару.
9.	price_in	float	-	Закупочная цена товара.

A.2 Таблицы хранения приходных накладных

В каждой записи таблицы pn хранятся атрибуты одной приходной накладной. Все кроме информации о позициях накладной. В таблице pns хранятся позиции накладной. Между таблицами существует связь типа "один-ко-многим"pn.id<=>> pns.pn_id

пояснения к полям таблицы “pn”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи в таблице (приходной накладной). Первичный ключ. Значения генерируются и заносятся программой. Пользователь не имеет доступа к этому полю.
2.	doctype	char	5	Символьное поле для хранения типа документа (не используется в настоящее время).
3.	docnum	int	-	Номер накладной, задаваемый пользователем. Может существовать несколько накладных с одинаковыми номерами
4.	docdate	date	8	Дата приходной накладной, задаваемая пользователем.
5.	client	char	80	Наименование покупателя. Это не ссылка, а именно наименование.
6.	summa	float	-	Сумма по накладной
7.	reason	char	60	Основание. Значение этого поля без изменений печатается в одноименном поле накладной.

пояснения к полям таблицы “pns”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи в таблице. Первичный ключ.
2.	pn_id	int	-	Ссылка на запись в таблице pn. Указывает на приходную накладную.
3.	tip	char	5	Наименование группы товаров, к которой принадлежит данный товар.
4.	name	char	50	Наименование товара. Это не ссылка, а именно наименование.
5.	kol	float	-	Количество закупаемого товара
6.	price	float	-	Закупочная цена товара
7.	summa	float	-	Сумма по товару

А.3 Таблицы хранения информации о контрагентах

“Справочник юридических и физических лиц” имеет одноуровневую структуру. Для его хранения используется таблица client.

Пояснения к полям таблицы “client”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор запись в таблице. Первичный ключ.
2.	name	char	80	Название юридического лица или ФИО физического лица
3.	otdel	int NOT NULL	-	Если поле имеет значение 1 - то контрагент является подразделением фирмы, ведущим самостоятельный учет, а не сторонней организацией. В противном случае поле имеет значение 0.
4.	phone	char	25	Телефон
5.	fax	char	25	Факс
6.	email	char	50	Адрес электронной почты
7.	address	char	80	Почтовый адрес
8.	bank	char	80	Название банковского учреждения
9.	bik	char	50	БИК
10.	korschet	char	50	Корр. Счет.
11.	rschet	char	50	Расчетный Счет.
12.	inn	char	50	ИНН
13.	okonh	char	25	ОКОНХ
14.	okpo	char	25	ОКПО
15.	prim	char	250	Примечания

А.4 Таблицы хранения журнала операций и информации об остатках

Пояснения к полям таблицы “oplog”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи в таблице. Первичный ключ. Задается программой. Пользователь не имеет доступа к этому полю.
2.	doc_id	int	-	Ссылка на документ. Идентификатор документа.
3.	tdate	date	8	Дата создания проводки.
4.	debit	char	5	Счет по дебету проводки
5.	kredit	float	-	Счет по кредиту проводки
6.	vsego	float	-	Сумма проводки
7.	kol	float	-	Количественное измерение учитываемого товара
8.	sbk_debit	char	255	Наименование объекта учета, например, товара по дебетовому счету
9.	sbk_kredit	char	255	Наименование объекта учета, например, товара по кредитовому счету
10.	prim	char	255	Примечания пользователя

Пояснения к полям таблицы "saldo"

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи в таблице. Первичный ключ. Задается программой. Пользователь не имеет доступа к этому полю.
2.	tdate	date	8	Дата, на которую расчитан остаток по счету. По одному и тому же счету в таблице одновременно хранятся остатки, расчитанные на разные даты. Даты, на которые расчитаны и хранятся остатки соответствуют датам проводок в журнале операций (таблица oplog).
3.	acn	char	5	Номер счета
4.	sbk	char	255	Наименование объекта учета
5.	vsego	float	-	Сумма остатка

A.5 Таблицы хранения информации о товарах. Справочник товаров

Справочник товаров предназначен для облегчения процедуры ввода данных в программу, путем замены по буквенной набивки наименования товара выбором его из списка. Накладные не содержат ссылок на позиции справочника товаров. В накладных хранятся сами наименования товаров, а не ссылки на них. Наименование товара может быть использовано вместе с названием группы, в которую входит товар, в качестве ключа для идентификации товара в справочнике. Однако, уникальность наименования внутри группы система не отслеживает.

Пояснения к полям таблицы “gr”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи в таблице. Первичный ключ. Задается программой. Пользователь не имеет доступа к этому полю.
2.	tip	char	5	Краткое название группы товаров (код группы)
3.	prim	char	50	Название группы товаров

Пояснения к полям таблицы “grs”

N	Имя поля	Тип	Длина	Пояснения
1.	id	int NOT NULL	-	Идентификатор записи в таблице. Первичный ключ. Задается программой. Пользователь не имеет доступа к этому полю.
2.	gr_id	int	-	Идентификатор группы товаров из таблицы gr
3.	tip	char	5	Краткое название группы товаров из таблицы gr
4.	name	char	50	Название товара
5.	custom	char	50	Номер таможенной декларации
6.	country	char	50	Страна происхождения товара

B Справочник функций API

B.1 Файл ananas_connect.c

```
#include <libintl.h>
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include "mysql.h"
  #include "ananas_connect.h"
  #include "rcread.h"
```

Макросы

#define _(String) gettext (String)

Функции

int **ananas_connection_get** (MYSQL parmysql)
Возвращает текущее соединение с MySQL.

int **ananas_connect** (char HN, char DBN, char UN, char Pass)
Устанавливает соединение с MySQL.

int **ananas_init** (ANANAS_INTERP interp)
Инициализирует библиотеку.

int **ananas_use_db** (char dbname)
Меняет текущую базу данных на ходу.

B.1.1 Макросы

#define _(String) gettext (String)

B.1.2 Функции

int ananas_connect (char *HN*, char *DBN*, char *UN*, char *Pass*) Устанавливает соединение с MySQL.

Имя хоста, базы, пользователя и пароль передаются через параметры.

Аргументы:

HN - имя хоста

DBN - имя базы данных

UM - имя пользователя

Pass - пароль

Возвращает:

0 - в случае успеха, 1 - в случае возникновения ошибки

int ananas_connection_get (MYSQL *parmysql*) Возвращает текущее соединение с MySQL.

Через параметр. Соединение хранится в статической переменной

Аргументы:

parmysql хандлер соединения

Возвращает:

0 - в случае успеха, 1 - в случае возникновения ошибки

int ananas_init (ANANAS_INTERP *interp*) Инициализирует библиотеку.

Считывает информацию из конфигурационного файла. Пытается прочитать файл .myrc и получить из него параметры для соединения с сервером. В случае успеха вызывает функуию **ananas_connect** (стр. 26) для установления соединения с сервером.

Возвращает:

0 - в случае успеха, 1 - в случае возникновения ошибки

int ananas_use_db (char *dbname*) Меняет текущую базу данных на ходу.

После успешного выполнения, текущей становится база, имя которой указано в пареметре.

Аргументы:

dbname - имя новой базы данных.

Возвращает:

0 - в случае успеха, 1 - в случае возникновения ошибки.

B.2 Файл ananas_doc.c

```
#include <libintl.h>
  #include <stdio.h>
  #include <stdlib.h>
  #include <string.h>
  #include "mysql.h"
  #include "ananas_doc.h"
  #include "ananas_connect.h"
```

Макросы

#define _(String) gettext (String)

Функции

int **ananas_doc_remove** (const char dt, char did)

Удаляет документ.

int **ananas_doc_tab_remove** (const char dt, char tid)

Удаляет строку табличной части документа.

int **ananas_doc_loadAll** (const char dt, ANANAS_DOC_RES result, const char flist)

Загружает указанные атрибуты всех документов.

int **ananas_doc_fetch** (ANANAS_DOC_RES result, ANANAS_DOC_ROW row)

Выбирает атрибуты документов.

unsigned long **ananas_doc_fields_lengths** (ANANAS_DOC_RES result)
Возвращает массив длин атрибутов документа.

unsigned int **ananas_doc_fields_num** (ANANAS_DOC_RES result)
Возвращает количетво выбранных запросом атрибутов.

int **ananas_doc_new_begin** (const char dt, char newid)
Добавляет новую накладную.

int **ananas_doc_tab_new_begin** (const char dt, char doc_id, char newid)
Добавляет новую строку в накладную.

int **ananas_doc_exists** (const char dt, char id)
Проверяет наличие документа в базе.

int **ananas_doc_tab_exists** (const char dt, char id)
Проверяет наличие строки табличной части документа в базе.

int **ananas_doc_tab_loadAll** (const char dt, ANANAS_DOC_RES result, const char id, const char flist)
Выбирает атрибуты всех записей табличной части указанного документа.

int **ananas_doc_tab_loadById** (const char dt, ANANAS_DOC_RES result, const char tab_id, const char flist)
Выбирает атрибуты одной записи табличной части указанного документа.

int **ananas_doc_loadById** (const char dt, ANANAS_DOC_RES result, const char id, const char flist)
Выбирает атрибуты одного документа.

int **ananas_doc_commit** ()
Закрывает ранее открытые транзакции.

int **ananas_doc_rollback** ()
Откатывает ранее открытые транзакции.

int **ananas_doc_update** (const char dt, char id, char fvs)
Обновляет значения полей документа.

int **ananas_doc_tab_update** (const char dt, char id, char fvs)
Обновляет значения записи табличной части документа.

B.2.1 Макросы

#define _(String) gettext (String)

B.2.2 Функции

int ananas_doc_commit ()

Закрывает ранее открытые транзакции.

int ananas_doc_exists (const char *dt*, char *id*)

Проверяет наличие документа в базе.

int ananas_doc_fetch (ANANAS_DOC_RES *result*, ANANAS_DOC_ROW *row*)

Выбирает атрибуты документов.

Атрибуты очередного документа выбираются из результатов ранее выполненного запроса. Возвращает через параметр массив значений атрибутов документа.

Аргументы:

result - переменная с результатами ранее выполненного запроса.

row - массив строк, в котором возвращвются значения атрибутов документа. Одна строка - один атрибут.

unsigned long ananas_doc_fields_lengths (ANANAS_DOC_RES *result*)

Возвращает массив длин атрибутов документа.

unsigned int ananas_doc_fields_num (ANANAS_DOC_RES *result*)

Возвращает количетво выбранных запросом атрибутов.

int ananas_doc_loadAll (const char *dt*, ANANAS_DOC_RES *result*, const char *flist*)

Загружает указанные атрибуты всех документов.

Список атрибутов передается через параметр. Атрибуты передаются в одной строке, разделенные запятыми.

Аргументы:

dt - тип документа.

flist - список атрибутов. Пример: "id, name, date, summa".

result - переменная-указатель на результаты запроса.

int ananas_doc_loadById (const char *dt*, ANANAS_DOC_RES *result*, const char *id*, const char *flist*)

Выбирает атрибуты одного документа.

int ananas_doc_new_begin (const char *dt*, char *newid*)

Добавляет новую накладную.

Возвращает идентификатор новой накладной Вы сами должны освободить пямять, выделенную под значение, аозвращаемое через параметр newid.

int ananas_doc_remove (const char *dt*, char *did*)

Удаляет документ.

Происходит удаление записей из двух связанных таблиц. Для сохранения целостности используется транзакция.

int ananas_doc_rollback ()

Откатывает ранее открытые транзакции.

int ananas_doc_tab_exists (const char *dt*, char *id*)

Проверяет наличие строки табличной части документа в базе.

int ananas_doc_tab_loadAll (const char *dt*, ANANAS_DOC_RES *result*, const char *id*, const char *flist*)

Выбирает атрибуты всех записей табличной части указанного документа.

int ananas_doc_tab_loadById (const char *dt*, ANANAS_DOC_RES *result*, const char *tab_id*, const char *flist*)

Выбирает атрибуты одной записи табличной части указанного документа.

int ananas_doc_tab_new_begin (const char *dt*, char *doc_id*, char *newid*)

Добавляет новую строку в накладную.

Возвращает идентификатор новой строки накладной Вы сами должны позаботится об освобождении памяти занятой переменной, через которую получен новый идентификатор.

int ananas_doc_tab_remove (const char *dt*, char *tid*)

Удаляет строку табличной части документа.

Происходит удаление записи из одной таблицы.

int ananas_doc_tab_update (const char *dt*, char *id*, char *fvs*)

Обновляет значения записи табличной части документа.

int ananas_doc_update (const char *dt*, char *id*, char *fvs*)

Обновляет значения полей документа.

B.3 Файл ananas_oplog.h

```
#include "mysql.h"
```

Макросы

#define **ANANAS_ANANAS_OPLOG_H** 1
#define **ANANAS_DOC_RES** MYSQL_RES
#define **ANANAS_DOC** MYSQL_ROW
#define **ANANAS_INTERP** MYSQL

Функции

int **ananas_oplog_add** (char newid, char doc_id, char tdate, char debit, char kredit, char vsego, char kol, char sbk_debit, char sbk_kredit, char prim)

Добавляет проводку в журнал операций.

int **ananas_oplog_remove_all** (char doc_id)

Удаляет проводки документа.

int **ananas_oplog_remove** (char id, char tdate, char debit, char kredit, char sbk_debit, char sbk_kredit)

Удаляет проводу.

int **ananas_oplog_loadBySbkDebit** (ANANAS_DOC_RES result, char value)

Поиск проводок.

int **ananas_oplog_loadBySbkKredit** (ANANAS_DOC_RES result, char value)

Поиск проводок.

B.3.1 Макросы

#define ANANAS_ANANAS_OPLOG_H 1

#define ANANAS_DOC MYSQL_ROW

#define ANANAS_DOC_RES MYSQL_RES

#define ANANAS_INTERP MYSQL

B.3.2 Функции

int ananas_oplog_add (char *newid*, char *doc_id*, char *tdate*, char *debit*, char *kredit*, char *vsego*, char *kol*, char *sbk_debit*, char *sbk_kredit*, char *prim*)

Добавляет проводку в журнал операций.

Вместе с этим вызывает корректировку остатков.

Аргументы:

newid (out) - идентификатор операции (проводки)

doc_id (in) - идентификатор документа, которому принадлежит проводка

tdate (in) - дата операции

debit (in) - счет по дебету

kredit (in) - счет по кредиту

vsego (in) - сумма операции

kol (in) - количество

sbk_debit (in) - субконто дебета

sbk_kredit (in) - субконто кредита

prim (in) - примечания

int ananas_oplog_loadBySbkDebit (ANANAS_DOC_RES *result*, char *value*)

Поиск проводок.

Возвращает проводки, в которых присутствует указанное дебитовое субконто.

Аргументы:

result (out) - результаты запроса

value (in) - субконто

int ananas_oplog_loadBySbkKredit (ANANAS_DOC_RES *result*, char *value*)

Поиск проводок.

Возвращает проводки, в которых присутствует указанное кредитовое субконто.

Аргументы:

result (out) - результаты запроса

value (in) - субконто

int ananas_oplog_remove (char *id*, char *tdate*, char *debit*, char *kredit*, char *sbk_debit*, char *sbk_kredit*)

Удаляет проводу.

Удаляет одну проводку с указвнным идентификатором. Вызывает обновление таблицы остатков.

Аргументы:

id - идентификатор проводки.

int ananas_oplog_remove_all (char *doc_id*)

Удаляет проводки документа.

Удаляет все проводки содержащие указанный идентификатор документа. Вызывает обновление таблицы остатков.

Аргументы:

doc_id - идентификатор документа.