Embedded Firebird & InterBase

У Firebird и InterBase кроме обычного сервера есть «встраиваемый» (embedded) вариант, который может использоваться (и на самом деле очень широко применяется, сотнями тысяч установок) для однопользовательских приложений.
Это сервер внутри dll (на Линуксе – so). Поскольку для работы с Firebird и InterBase приложениям обычно требуется клиентская библиотека (за исключением драйверов типа Java, которые работают напрямую с сервером), для работы с embedded обычно требуется использовать dll embedded вместо клиентской dll.
В результате приложение может, как обычно, работать с удаленным сервером через сеть, и может работать с локальными базами данных без установленного экземпляра сервера. Для приложения разница будет только в формате строки коннекта к БД, а дальше клиентская библиотека будет работать или с удаленным сервером, или с локальной базой данных. Других отличий для приложения не будет, embedded обладает абсолютно той же функциональностью, что и обычный сервер.

Оглавление

Embedded Firebird & InterBase 1
Немного истории 2
Windows, Linux? 2
Где скачать 2
Платформы и разрядность 3
Инструменты командной строки в комплекте 3
Как работает приложение с Embedded 3
Архитектура Embedded и взаимодействие приложений 4
Права доступа Firebird 6
Права доступа на Windows и Linux 6
Разработка и отладка 7
Специфическое применение 7
Странное использование 7
Шифрование баз и Embedded 8
Embedded и Embedded SQL? 8
Где используется Firebird Embedded 8

Немного истории

Первая версия встроенного сервера появилась в Yaffil (клон InterBase 6.0 после публикации его исходных кодов в 2000 году), и называлась Yaffil Personal. Собственно, задел для такой возможности существовал еще в InterBase 4 – там небольшой по размеру (92кб) процесс сервера ibremote.exe на самом деле вызывал gds32.dll (1мб), где и находился код реального сервера, работающего с БД.
Авторы Yaffil подправили код, и в результате получили версию embedded.

Проект Firebird тоже выпустил встраиваемый вариант под названием Firebird Embedded, это произошло с версии 1.5.1, примерно в июле 2004 года.

У InterBase встраиваемая версия вышла только в 2008 году (это InterBase 2009) под названием IBToGo. Поскольку сам InterBase платный, вариант ToGo стоил 60 долларов за 1 лицензию, не имел ограничений, и существовал только для Windows. Затем, в мае 2013 года, был выпущен дополнительный бесплатный вариант под названием IBLite, но его функциональность была урезана – работал он только на одном ядре, не более 1 коннекта, отсутствует «сильное» шифрование коннекта и БД, и размер БД не может превышать 100мб.
Выпуск IBLite был приурочен к кроссплатформенной разработке, и в основном предназначался для использования на мобильных устройствах iOs (Delphi XE4) и Android (Delphi XE5).

Windows, Linux?

В дальнейшем речь будет идти именно о Windows. На Linux в отношении клиентской библиотеки всё точно так же (разве что расширение - .so а не .dll).
(ну не «всё», а «почти всё», но если вы пишете клиентские приложения для Linux, вы и так это знаете)

Где скачать

Yaffil теперь уже нигде не взять, его разработка была прекращена в 2003 году.

Firebird Embedded – для 1.5 и 2.x это отдельные архивы на sf.net/projects/firebird и github.com/FirebirdSQL со словом embed в имени архива, например Firebird-1.5.6.5026-0_embed_win32.zip, https://github.com/FirebirdSQL/firebird/releases/download/R2_5_9/Firebird-2.5.9.27139-0_Win32_embed.zip, https://github.com/FirebirdSQL/firebird/releases/download/R2_5_9/Firebird-2.5.9.27139-0_x64_embed.zip.
У Firebird 3.0/4.0/5.0 отдельного архива Embedded нет (о причинах будет рассказано дальше), нужно скачивать просто архив zip нужной разрядности со страницы https://firebirdsql.org/en/firebird-3-0/ , https://firebirdsql.org/en/firebird-4-0/ и https://www.firebirdsql.org/en/firebird-5-0 .

IBLite и IBToGo отдельно скачать нельзя, они распространяются вместе с Delphi, C++Builder или RAD Studio – после установки можете заглянуть в папку C:\Users\Public\Documents\Embarcadero\Studio\19.0\CatalogRepository (здесь 19 – внутренний номер RAD Studio 10.2, если у вас другая версия, ищите соответствующий номер у себя), далее найдете папку с названием вроде InterBase_Redist-19.0.31059.3231, в ней уже будет архив Interbase_redist.zip, в котором находятся IBLite/IBToGo для Windows, Android, OSX, Linux и т.д.

Да, разделения на IBLite и IBToGo в этих папках нет, потому что это один сервер, а отличия есть только в файле лицензии.

Платформы и разрядность

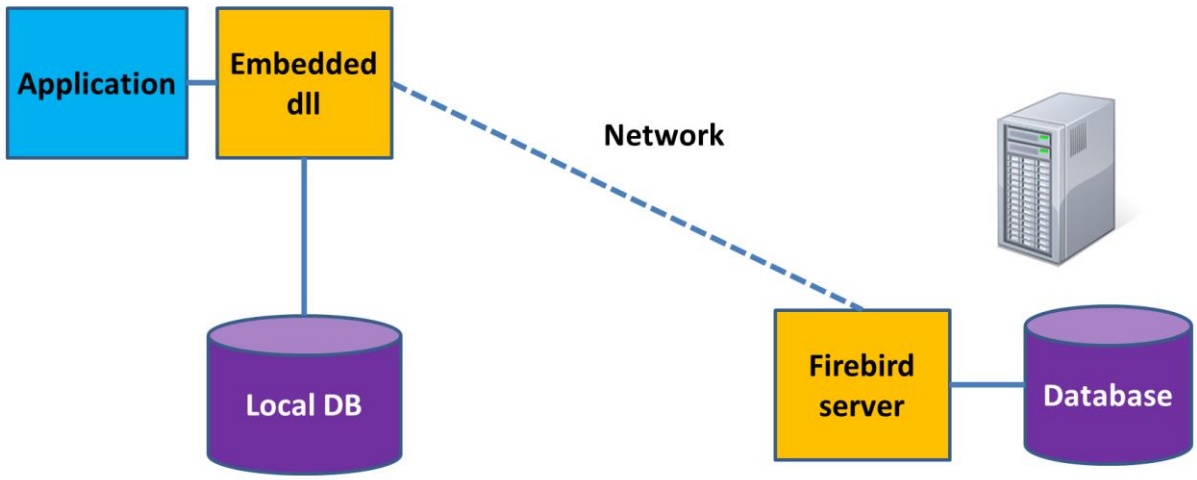
Firebird 1.5 Embedded – Win32
Firebird 2.5 Embedded – Win32/64, Linux 32/64.
Firebird 3.0/4.0/5.0 Embedded – Win32/64, Linux 32/64, есть тестовый билд для iOS, также можно использовать тестовые билды для MacOSX на iOS.
IBLite/IBToGo – Win32/Win64, OSX32, iOs 32/64, Android, Linux 32/64

Инструменты командной строки в комплекте

Имеются в виду утилиты - gbak, gfix, gstat - их нет в архивах с Embedded 1.5 и 2.x с самого начала появления Embedded (Firebird 1.5). А в 3.0/4.0/5.0 архив сервера содержит все необходимые файлы.
Собственно, для Embedded управление бэкапом, рестором и прочим предполагается «изнутри», т.е. при помощи Services API (например, https://www.ibase.ru/ibx#servapi). Но может потребоваться выполнять сервисные операции извне приложения. Поэтому взять утилиты командной строки для 1.5/2.x можно из архива сервера соответствующей разрядности, а в архивах 3.0/4.0/5.0 и так есть полный комплект.

Как работает приложение с Embedded

Основной смысл Embedded в том, что это dll с именем, идентичным штатной клиентской библиотеке. Например, в Firebird 2.5 достаточно переименовать fbembed.dll в fbclient.dll – и мы как-бы получаем два в одном: и клиентскую библиотеку, и сервер внутри dll (fbembed.dll это и так fbclient.dll с включенным туда кодом сервера).



Разрядность Embedded, разумеется, должна соответствовать разрядности приложения – 32-разрядное приложение не может загрузить 64-разрядную dll, и наоборот
(напомню, что разрядность клиентской библиотеки и обычного сервера может быть разной, поскольку сетевой протокол «разрядности» не имеет).

В результате – запускаем наше приложение (exe), оно загружает fbclient.dll (или gds32.dll), и при локальном коннекте (без имени сервера, например c:\db\data.fdb) сервер в DLL начинает работать как обычный сервер, но внутри (в адресном пространстве) нашего приложения.

А если указан сетевой коннект, например localhost:c:\dir\data.fdb, то он из Embedded перенаправляется на сервер (если тот установлен и работает) точно так же, как и в случае обычного fbclient.dll.

! начиная с Firebird 3 embedded работает несколько по другому: fbclient.dll теперь всегда только клиент. Он либо обращается к серверу по сети, либо при embedded-коннекте пытается загрузить движок Firebird – engineNN.dll, где NN – версия формата базы данных конкретной версии Firebird (см. https://www.ibase.ru/prevver/). Так что разрядность приложения, fbclient.dll и dll движка должна быть одинаковой. Подробнее см. далее «Архитектуры…».

В отношении приложения+Embedded есть несколько моментов:

- если ваше приложение содержит ошибки, и периодически «падает», то это означает падение «сервера» embedded, а значит, **может привести к повреждению базы данных**.
- клиентская часть, даже с включённым Embedded, не имеет «внешнего слушателя» по tcp, поэтому ваше приложение с Embedded не может выступать в роли обычного сервера Firebird (или InterBase). Но если вы в приложении сами реализуете связь по сети с какими-то другими приложениями, по своему собственному протоколу – то да, может.
- в зависимости от настроек firebird.conf или ibconfig, Embedded может потребить много памяти, в результате чего, если приложение 32битное, может произойти переполнение памяти приложения (не более 2 гигабайт), и приложение терминируется с ошибкой.

Архитектура Embedded и взаимодействие приложений

Firebird до сих пор поддерживает 3 архитектуры – Classic, SuperClassic, SuperServer (у InterBase есть только SuperServer начиная с версии 7.0).
Если вы не очень понимаете разницу архитектур, я рекомендую посмотреть видео с подробным объяснением. В данный момент нас интересует в основном то, как эти архитектуры работают с файлом БД.
Classic и **SuperClassic** открывают файл БД в shared-режиме, то есть, 2 и более процессов Classic или SuperClassic могут одновременно работать с одним файлом БД, синхронизация осуществляется через менеджер блокировок.
SuperServer открывает файл БД в эксклюзивном режиме, в противном случае 2 процесса SuperServer могли бы испортить файл БД, т.к. между ними нет синхронизации. Поэтому при такой попытке вы получите сообщение об ошибке.
(У InterBase есть только SuperServer, и начиная с XE7 второй экземпляр сервера зачем-то открывает БД в режиме read-only).

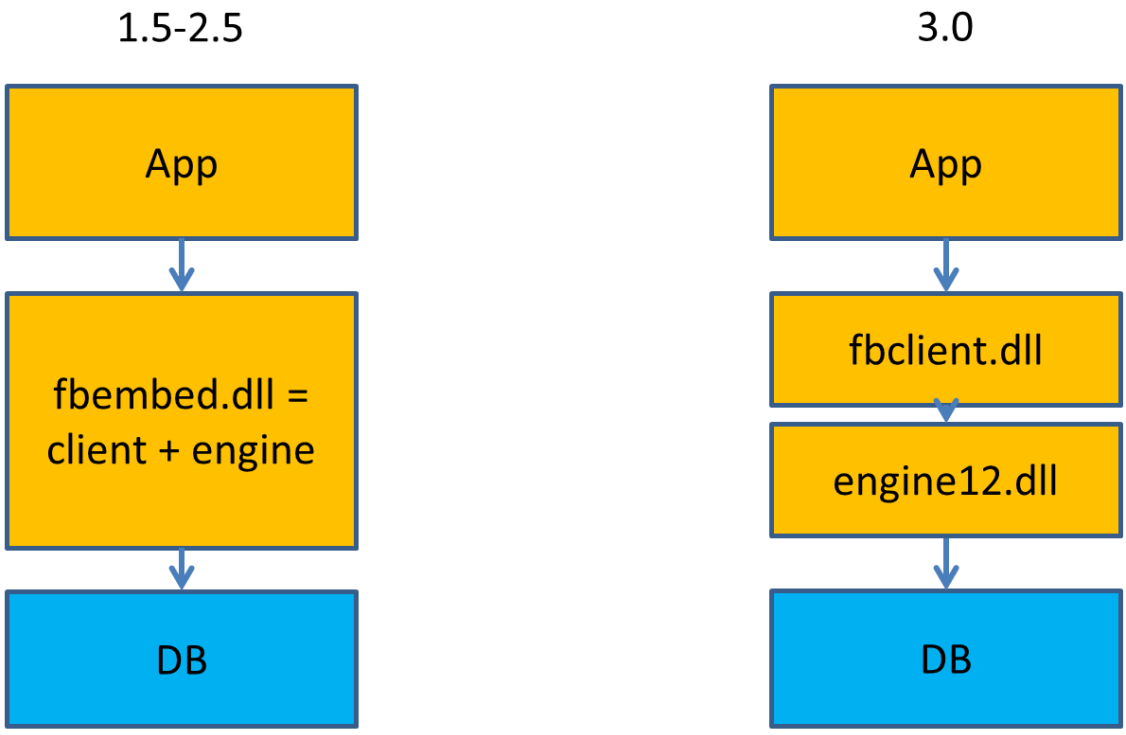
Поскольку в случае Embedded сервером является экземпляр приложения, то несколько приложений будут работать с БД в полном соответствии с архитектурой Firebird

- Firebird 1.5, 2.0, 2.1 Embedded – SuperServer
- Firebird 2.5 Embedded – SuperClassic
- Firebird 3.0/4.0/5.0 Embedded – Classic/SuperClassic или SuperServer, в зависимости от настройки в firebird.conf (по умолчанию SuperServer)

Если для Embedded 3.0/4.0/5.0 указать режим Classic, то на самом деле будет режим SuperClassic, потому что для второго коннекта к той же БД из вашего приложения для имитации Classic никто запускать второй экземпляр вашего приложения не будет.

Исходя из изложенного выше, в 1.5, 2.0 и 2.1 два приложения не смогут работать с одной БД. В Firebird 2.5 – смогут, хоть десять. А в 3.0/4.0/5.0 – смогут или нет в зависимости от настроек.

Кстати, даже при первоначальной работе с сервером Firebird 3.0/4.0/5.0 люди стали сталкиваться с проблемой блокирования файла БД. Например, подключаются к базе через isql, указывая локальный коннект (получается Embedded-соединение в режиме SuperServer, который по умолчанию указан в firebird.conf), а затем при сетевом или локальном коннекте к этой же БД – получают ошибку.
Раньше (до 3.0/4.0/5.0) локальный коннект без Embedded осуществлялся по протоколу xnet (к серверу, разумеется), и не приводил к таким проблемам.
Кроме того, если в 1.5/2.5 файл конфигурации firebird.conf был не совсем обязательным, то для 3.0/4.0/5.0 он совершенно обязателен, т.к. как минимум содержит параметр архитектуры сервера ServerMode = Super | SuperClassic |Classic
(а кроме того параметры шифрования, аутентификации, и т.д.)



В Firebird 3.0/4.0/5.0 локальный коннект сразу использует режим Embedded (это видно из картинки). И для использования xnet вместо embedded этот протокол необходимо явно указывать в строке коннекта – например, xnet://c:\db\data.fdb.

! Клиентская библиотека, начиная с 3.0, при локальном коннекте (без имени сервера) сначала пытается сделать embedded-подключение, а если не получилось – переходит на xnet, неявно. Если в этом случае есть локальный сервер, и файл базы уже не заблокирован Embedded в режиме SuperServer, то подключение по xnet может получиться. Если локального сервера нет, то подключения по xnet не произойдет, просто некуда подключаться. Например,
D:\Firebird3>isql e.fdb -user SYSDBA
Statement failed, SQLSTATE = 08001
I/O error during "CreateFile (open)" operation for file "e.fdb"
-Error while trying to open file

! engine12.dll – это библиотека сервера 3.0, которая находится в подпапке /plugins. У 4.0 это engine13.dll, у 5.0 – engine13.dll, для которой базовой является ODS 13.1 (структура БД).

Так что, если хотите, чтобы с базой могли одновременно работать несколько приложений на одном компьютере – сразу укажите для Embedded в firebird.conf параметр **ServerMode = SuperClassic**.

Права доступа Firebird

Понятно, что приложение+embedded это «сам себе сервер», поэтому **для Embedded права доступа не проверяются** (логин к базе идет «в обход» securityN.fdb), и **выданные в БД права на объекты не работают**.

Однако имя пользователя при логине всё равно нужно указывать. Это может быть SYSDBA, или даже ABCD.

D:\Firebird3>isql e.fdb -user ABCD
Database: e.fdb, User: ABCD
SQL>

Пароль можно указать (-password whatever), но он игнорируется.

! Из этой функциональности происходит решение по созданию пользователя SYSDBA, которого нет при установке Firebird 3.0 и выше инсталлятором. Первое действие – подключиться к базе employee (или любой другой) в режиме embedded:
>isql employee –user SYSDBA

Далее – создать SYSDBA командой
SQL> create user SYSDBA password 'here_is_your_password';
SQL> commit;
SQL> quit;
В результате в security?.fdb появится пользователь SYSDBA с указанным паролем, и к серверу можно будет подсоединиться по сети. Понятно, что для embedded выполнять эту операцию не нужно.

Если вас мучает необходимость указания имени пользователя при использовании утилит командной строки с Embedded, вы можете в системе создать переменную ISC_USER, и дать ей значение SYSDBA. Embedded будет для логина брать значение этой переменной автоматически.

Права доступа на Windows и Linux

Штатно СУБД Firebird сама работает с файлом БД, поэтому права на доступ к файлу БД должны быть доступны СУБД Firebird. Это не файл-сервер, а клиент-сервер, поэтому пользователям давать права на доступ к БД не нужно.

На Windows по умолчанию Firebird устанавливается и работает от имени пользователя LocalSystem, у которого права доступа как минимум совпадают с администратором этого компьютера. Поэтому никаких дополнительных настроек не требуется.

Но, на Linux при установке Firebird сразу создается пользователь firebird, под которым и работает процесс (или процессы) СУБД Firebird, и который должен иметь права на доступ к базе данных (или в папки бэкапов, где бэкап делается через Services API – опцией –se или через утилиту fbsvcmgr).

На Windows, если вы поменяете пользователя у службы Firebird, также придется давать соответствующие права на папку с БД.

Теперь, если вы на этой же конфигурации пытаетесь работать с Embedded, то получается, что
- у вашего приложения есть права какого-то пользователя

- приложение, загружая движок БД получает те же права
- движок пытается применить те же права к базе данных

И если права не соответствуют, то будет ошибка доступа. Так что, лучше следовать таким правилам:
- если предполагается работа только через Embedded, не запускайте сервер Firebird, и не конфигурируйте права доступа к БД для пользователя firebird, иначе будут конфликты доступа
- если предполагается работа только с нормальным сервером, не используйте на нём embedded-подключения, иначе придется специфически настраивать права доступа к БД.

Разработка и отладка

Использование Embedded при разработке и отладке крайне неудобно. Вам сначала придется подложить файлы Embedded в папку со средой разработки, чтобы в ней можно было работать с БД в дизайн-тайме, и еще подложить файлы embedded в папку с компилируемым приложением. При этом, если embedded работает в режиме SuperServer, при первом коннекте (из среды разработки, например) база заблокируется, и второй коннект из отлаживаемого приложения уже не пройдет. Поэтому придется переключать режим сервера в Classic/SuperClassic в firebird.conf во всех местах расположения embedded.
Кто-то может решить положить embedded в одну папку, и настроить в системе PATH, чтобы и среда и приложение искали его в одном месте. Но тут совершенно элементарно про это забыть, и получить много удивительных проблем, когда вам понадобится протестировать или другую версию embedded, или поработать с другой версией сервера, и т.д.

Поэтому, использовать Embedded для разработки и отладки просто не надо. Используйте обычный сервер, с коннектом по tcp (например localhost:c:\dir\data.fdb). Как только приложение будет отлажено, можно будет проверить его «распространяемость» убрав сетевой коннект из компонент или настроек, и подложив Embedded рядом с exe. Вам всё равно придется делать такую проверку на отдельном компьютере (для чистоты эксперимента), так что это всё можно сделать и после отладки.

Специфическое применение

Потоковая конвертация баз формата 2.5 в формат 3.0 на примере двух Embedded – 2.5 и 3.0. Описание здесь
https://habr.com/ru/post/445204/

Странное использование

Иногда у разработчиков появляются странные идеи по использованию Embedded. Это и подключение как dll/so к разным веб-серверам, или просто многопользовательские приложения, и т.д. Делать этого совершенно не нужно по одной простой причине – такие решения не масштабируются, и потребляют гораздо больше памяти, чем ожидалось. Гораздо правильнее сразу использовать обычный сервер Firebird, настроить его как нужно, а при необходимости (например, нехватке ресурсов) – вынести сервер Firebird с базой на отдельный компьютер или виртуальную машину.

Шифрование баз и Embedded

Начиная с Firebird 3.0 база данных может быть зашифрована. Если для обычного сетевого сервера Firebird ключ шифрования может оставаться на сервере (к которому мало кто имеет доступ), то для Embedded ключ шифрования должен быть только в вашем приложении, иначе такая конструкция теряет смысл.

При этом вы лишитесь возможности использовать утилиты командной строки gbak/gfix/gstat, т.к. внедрить в них ключ шифрования невозможно (возможно, только если вы можете сами скомпилировать эти приложения из исходных текстов). Поэтому функционал этих утилит, если он вам нужен, придется использовать через Services API.

! ООО «Айбэйз» поставляет готовый плагин и примеры по шифрованию баз и использованию ключей в приложениях – Firebird Encryption Plugin Framework.

Embedded и Embedded SQL?

Было когда-то такое заблуждение, что мол, в Embedded нужно/можно использовать Embedded SQL (или наоборот). **Никакой взаимосвязи.** Embedded SQL – это специальные конструкции SQL, которые обрабатываются препроцессором (утилитой gpre), и превращаются в код, комбинирующий C/C++ с обычным SQL.

Сейчас про Embedded SQL уже практически никто не помнит, т.к. он используется исчезающе-малым количеством компаний (в РФ – 1-2, не более).

Где используется Firebird Embedded

К сожалению, оценить количество использований Firebird Embedded невозможно, т.к. количество скачиваний архивов с sourceforge или github никак не проецируется на реальное количество установок. Инсталлятора у embedded или zip нет, а разработчики приложений о своих достижениях обычно не сообщают.

Могу разве что сказать, что известная программа CheckPFR (отчетность в Пенсионный фонд) использует Firebird 2.5.

В общем, там, где нужны однопользовательские приложения с базой данных – Firebird Embedded является отличным выбором. Embedded идентичен отдельному серверу, приложение не требует переделки при его переводе на работу с обычным сервером, не требует много ресурсов, и т.д.

Вопросы? Пишите на support@ibase.ru