Things to consider when migrating to Firebird 3

By Carlos H. Cantu (Oct, 2017)

www.firebirdnews.org
www.firebase.com.br

Backup/Restore is now a requirement!

Moving to a new Firebird version was always considered a very easy process. Many people used to just uninstall the old Firebird version and install the new one. Some of them did not even bothered to make a backup/restore of the databases, to update the ODS to the newest version. This is a bad idea! Some new features can be used only if the ODS and client versions are up to date.

Keep in mind that no matter to what Firebird version you are moving to, you should always do **backup/restore** and update the client library (*fbclient*), both in the server as well in the terminals (clients). **For moving to Firebird 3, this is not just an advice, it is a requirement**!

Firebird 3 cannot connect to databases with older ODS (ODS < 13). In other words, it will not connect to databases created in past versions of Firebird. As I state in my *Migration Guide to Firebird 3* book (MGTFB3), you are required to do a backup/restore using gbak to update the ODS. **The backup must be done while your old Firebird version is still running.**

Consider that gbak is a "normal" client/server application, meaning that it depends on Firebird to connect to the database and do its job, which mostly consists in reading all the database structure and data, and save it to the backup file. If you already installed Firebird 3 over your previous version, gbak will not be able to connect to the databases with older ODS.

Several changes to User Management

So, moving to Firebird 3 requires a backup/restore, but that's not all! Firebird 3 major achievement is to offer a SuperServer architecture that is fully scalable, while still providing a shared cache among the connections. But developers implemented several other long waited features, and some of them required a major rework and/or introduction of concepts. For example, in Firebird 3, it is possible to have different users with the same name! *Uh? What? How can this be?* That is possible because Firebird 3 introduces the concept of *User Manager Plugins,* where each plugin has its own users.

When you create a new user in FB 3, you can specify what plugin you want to use. So, you can have JOHN with password 1234 created in one plugin, and JOHN with password 5678 created in another plugin. Yes, this can be a bit confusing, and often may lead to users hitting their heads against the wall, especially when they do not carefully read the documentation and are not aware of the new concepts in user management.

Other changes in this area are the fact that users now can be created inside the main database (*local users*) or in any other security database you may want to have, and the user maintenance (creating/delete/updating users) can be done using SQL statements. You also must remember that there is no way to migrate your previous security database (security*x*.fdb) to Firebird 3, keeping the existing users with their currently passwords, which, by the way, now supports up to 255 bytes in length, instead of the previous 8 characters limitation.

The *security_database.sql* script that comes with Firebird 3 installation may help you to create your existing Firebird users in Firebird 3. The script reads the existing users on the old security database, retrieving the names and attributes, and creates those same users in Firebird 3, with a random password generated by the internal gen_uuid function. When executed, it will display the names of the users and their generated passwords.

I dedicated a full chapter in the MGTFB3 book, just to talk about user management in Firebird 3.

Security enhancements

Firebird 3 also brings enhanced security. The default is to have all network traffic encrypted, and there is an option to encrypt the database file, although Firebird does not come with a default plugin for database file encryption. You can create one by yourself, or buy one from third party companies, like IBSurgeon or IBPhoenix.

Don't fool yourself thinking that the new "*local database users*" concept will protect you from having your data stolen if someone is able to copy your database file and take it to another server. Only a well-done encryption plugin can protect you in such case.

In the MGTFB3 book, I describe the areas that requires your attention when trying to protect your databases. FB 3 default installation comes with a C++ example code of a *dbCrypt* plugin, and you can use it as a base for implementing your own solution. Please do not compile the provided example "as is" and use it in production, since it actually does not offer any real protection. It is just an example!

Wire Protocol

Another improved area in FB 3 is the *Wire Protocol*. It is well known that the protocol was always very "chatty", meaning that there is a lot of conversation between the *client* and the *server*. In a local network, this is mostly unnoticed, but if you have pure C/S applications accessing remote database servers over the internet - or over any other high latency network - you probably already suffered with the bad performance imposed by the chatty protocol.

It is true that Firebird 2.1 already introduced some improvements in this area, but Firebird 3 took this to another level, and now, if you don't use blobs too much and is careful when writing your *selects*, it is perfectly possible to have pure C/S application connecting remote servers with good performance. By the way, the enhancements in the FB 3 wire protocol was sponsored by donations collected in the 9th Firebird Developers Day Brazilian Conference.

Checking your metadata

Last, but not least, you must pay attention in your existing metadata. The required backup/restore will not recompile your *triggers* and *procedures* sources, and even a successful backup/restore may lead to bad surprises during the real use of the database.

You must check for new *reserved words* and for possible changes in the query's PLANs. Firebird 3.0.2 already fixed some optimizer problems that made the performance of certain queries in Firebird 3.0 worse than in 2.5, so my advice is to start with Firebird 3.02 and skip 3.0 and 3.01.

An easy way to check if your metadata source is 100% compatible with Firebird 3 is to extract the metadata into a SQL script (*isql -x*), and run this script in Firebird 3 (*isql -i*) to generate a new database. If it breaks, you will need to fix the problems before going into production.

The most common problem relates to *variables* or *objects names* that became *reserved words* in FB 3. Of course, you can just put them in *double quotes*, but you should think carefully before doing this, since it will make them case-sensitive, meaning that you may end up rewriting a lot of existing queries and will always need to write new code referring them with the correct case, forever!

Conclusion

As you saw, a smooth migration to Firebird 3 requires a lot of attention in different areas. If you are lucky enough, a backup/restore will be the only thing need. But if you are smart and don't like bad surprises, spending some time to check all the dangerous points, before going into production, will save you from having to deal with mad customers or furious users.

In spite of the topics discussed in this article, there are many other subjects that requires your attention, especially if you want the transition to Firebird 3 to go as smooth as possible. That is why I wrote an entire book about this. The **Migration Guide to Firebird 3** can save you from innumerous hours of research and headache. You can buy the PDF version or the printed version at https://www.firebirdnews.org/migration-guide-to-firebird-3/