Delphi Summit
June 2024 | Amsterdam

The *Global* **Delphi** Summit

Organized by

embarcadero®

GDK software

barnsten

Synchronize your databases

Patrick PREMARTIN

https://gravatar.com/patrickpremartinfr

https://developpeur-pascal.fr

BEGIN

INTERFACE

- What do we want to solve?
- What solutions currently exist?
- Why not use them?
- The original idea
- My solution
- Open source version
- Examples of uses

IMPLEMENTATION

We live in a connected world where office and tertiary employees in general no longer want to work in the office.

End users of our software, especially individuals, are nomadic and want to access their data on different devices from anywhere.

Delphi allows you to make desktop software, mobile apps and websites but it doesn't solve everything.

IMPLEMENTATION

When we offer our users freedom of movement, we force them to stay connected to the Internet. In practice this is not possible anywhere on the planet or above. And it can be very expensive depending on where we are.

Technical solutions are usually data replication, client/server work or a mixture of both.

I put the web solutions in the client/server box.

IMPLEMENTATION

Some databases include systems for synchronizing changes in the form of requests (SQL cache) or data (such as InterBase).

The disadvantage of these solutions is often at the level of primary keys: if several users make additions to the same table and these tables are linked to other ones it can get stuck.

IMPLEMENTATION

Software running with a data server depends on a connection to it to work.

On a local business network it works well as soon as we get out it can get caught.

An absence of the Internet prevents the use of remote software from requiring a direct connection to the database server or from passing through an intermediate program such as DataSnap.

IMPLEMENTATION

Faced with these problems, companies have developed their synchronization solutions for one or more projects.

These solutions are rarely public, shareable or usable by others.

IMPLEMENTATION

In 2017 I worked with Christian Hubert-Hugoud on the Delphi redesign of his Intelli7 software.

IMPLEMENTATION

Among the needs and constraints were:

- one database per client
- one or more users per database
- access to the same features from a computer or tablet
- access from anywhere, with or without Internet connection
- possibility to modify client bases from the server or other software than Intelli7

IMPLEMENTATION

We have designed a blending replication and client-server solution in the form of API.

The Intelli7 software was developed in Delphi under FireMonkey and used some components of TMS Software to display graphics and export them in PDF.

The software uses a local SQLite database to store a file system, NoSQL data in JSON and parameters in the form of key/value pairs.

IMPLEMENTATION

Each instance of the software has a copy of the database and works on it locally without worrying about others.

A process regularly synchronizes data when an Internet connection is available.

The fact that the database is synchronized is completely transparent for the software and users.

IMPLEMENTATION

On the server side we worked in PHP with SQLite databases. It was easier to manage than a multitude of MySQL databases to ensure that no customer would ever have access to the data of others.

An API was created for synchronization.

It does not depend on the development languages or the database engines used.

IMPLEMENTATION

A Delphi Developer-oriented Intelli7 demonstration is available at https://vimeo.com/918818000/9c4ab12439

A study case is available at https://github.com/DeveloppeurPascal/DelphiSummit2024-Synchronize-Your-Databases

To find out more about the software and the Intelli7 method, visit https://intelli7.com

IMPLEMENTATION

The solution developed for Intelli7 has evolved over time. I am inspired by it for other applications until I get the version you will be able to use in your projects.

The basic principle is to minimize impact on existing software while providing them with the possibility to switch to multi-user/multi-device/multi-location mode.

IMPLEMENTATION

You need a server that users can access from time to time :

- on the Internet for total mobility
- on a company network (locally or via a VPN), depending on your security requirements.

Replication is based on fields in your database. A change in its structure will be necessary.

IMPLEMENTATION

For security reasons, you must take all necessary steps to protect your databases and software.

If you authorise use outside your premises, as this system is based on a copy of the data, take the necessary steps to encrypt it and lock access to your software (username, password, 2FA)!

IMPLEMENTATION

Minimise the impact of stolen computers, smartphones or tablets on the integrity and confidentiality of your data.

The same applies to devices that remain on your users' premises, including servers.

IMPLEMENTATION

I've called my synchronisation solution Table Data Sync. It's a simple client/server API.

A Delphi implementation is available at https://github.com/DeveloppeurPascal/TableDataSync4Delphi

Other implementations for other languages will be put online at https://github.com/DeveloppeurPascal?tab=repositories&q=tabledatasync&type=&language=&sort=

IMPLEMENTATION

There are 4 steps to synchronisation:

- opening a session
- receive changes saved on the server
- sending local changes to the server
- closing the session

IMPLEMENTATION

The connection allows you to define access codes and tell the server which database you want to work on.

This data is sent using http or https.

It's up to you to encrypt them for an extra level of protection.

IMPLEMENTATION

Data is received from the server in relation to a version number stored on each record for each of the tables to be synchronised.

IMPLEMENTATION

Sending local data to the server selects all records that have been added, modified or deleted since the last synchronisation in each table to be synchronised.

IMPLEMENTATION

Closing the connection releases the session on the server side and enables the end of synchronisation to be detected on the client side.

IMPLEMENTATION

Physical deletions are detected by means of an auxiliary table that the program will have to fill in when a record is deleted somewhere.

Logical deletions are handled in the same way as other modifications, since they are generally processed in the form of a field in the tables concerned.

IMPLEMENTATION

Synchronisation conflicts are handled on the client side. The library provides events to manage them automatically or manually.

If they are not handled, the data is overwritten according to the last modification date.

IMPLEMENTATION

For this to work, you need to add fields to the tables you want to synchronise:

- each identifier must be doubled to have its number assigned on the server side (as a primary key as well as a reference)
- a Boolean field (or equivalent) must be used to define whether a record has been added or modified since the last synchronisation

IMPLEMENTATION

For this to work, you need to add fields to the tables you want to synchronise:

- an integer field to be used as the version number of each record in its table
- a date/time field to be updated each time a change is made

IMPLEMENTATION

It is recommended that you add unique indexes using these new fields in order to speed up the synchronisation stages and avoid any negative impact on the response times of your software.

If you make physical deletions from your tables, you should add a table to keep a record of these deletions and pass them on to other installations of your software.

IMPLEMENTATION

You then need to add triggers to your tables or process these operations using code (but be careful not to forget any):

- an insertion or update must position the sync field and change the update date
- a physical deletion must add the unique key to the deletions table

IMPLEMENTATION

In the current version, the Delphi library only supports integer primary keys or combinations of integer keys.

Each table to be synchronised must be accessible with a unique index on which the synchronisation can be based.

IMPLEMENTATION

The database on the client side must have the same structure as on the server. This means updating it at the same time on both sides or managing different versions depending on your changes to its structure.

The current way of working is not designed for large volumes of modifications per table. It would consume too much bandwidth.

IMPLEMENTATION

Table Data Sync is best suited to ‘nomadic’ users who work on their own data and need to synchronise with other users from one or more devices.

Examples include sales staff, delivery personnel, teleworkers, ‘virtual’ service providers or assistants and, of course, private users of software for which you provide hosting for the synchronised databases.

IMPLEMENTATION

Table Data Sync is first and foremost an API running on the HTTP/s protocol so that it can be used everywhere.

My Delphi or PHP implementation of the client library and server may not work for you. There's nothing to stop you modifying them and adapting them to your needs while keeping the ‘other side’.

IMPLEMENTATION

There is nothing to stop you transferring only part of the data to each workstation, depending on the identifiers used when the session was opened.

You can also have several programs working with the same synchronisation server but using only part of the data (which is also recommended to avoid unnecessary duplication of information, which would increase the risk of leaks).

IMPLEMENTATION

It is possible to have a server that is independent of the database structure if you don't need to process data on the server side.

IMPLEMENTATION

The version of Table Data Sync available at https://github.com/DeveloppeurPascal/TableDataSync4Delphi is distributed under the AGPL v3 licence.

You can use it for both your personal and professional projects.

If the AGPL licence doesn't apply to you, I also sell classic 'developer licences', 'site licences' and 'company licences'.

IMPLEMENTATION

As this involves manipulating data without any visibility for the end user, carry out tests before putting your projects into production.

Make backups of your users' data in your software and on the servers so that you can go back if you detect an anomaly one day.

The use of my Table Data Sync implementations is under your control. No guarantee is provided.

IMPLEMENTATION

I use this version of the library in small projects but haven't had the opportunity to deploy it widely. Everything works fine for my use.

There are undoubtedly problems that I haven't yet detected. Remain suspicious and report any anomaly or weirdness to me.

IMPLEMENTATION

Between two Twitch streams or open source developments ‘for the community’ (and for me) I also provide consulting services.

If you need advice to check the relevance of my solution or to implement it in your projects, don't hesitate to talk to me. I'll see what I can or can't do to help.

IMPLEMENTATION

I've developed several simple examples to illustrate this presentation. You can find them on this GitHub repository : https://github.com/DeveloppeurPascal/DelphiSummit2024-Synchronize-Your-Databases

A WebBroker server handles synchronised databases for client projects. The server runs on Linux.

The client programs have been tested on Windows, macOS, iOS and Android.

FINALIZATION

Table Data Sync can make your life easier and your users' lives easier.

The library for Delphi is available at https://github.com/DeveloppeurPascal/TableDataSync4Delphi

The library for PHP is available at https://github.com/DeveloppeurPascal/TableDataSync4PHP

FINALIZATION

To contact me, please use one of my websites or the links on https://gravatar.com/patrickpremartinfr

I am also available between sessions during the conference.

END.