Using SQLyog Enterprise to Effectively Synchronize MySQL Databases

By Peter Laursen and Quy Ton

Note: This document contains information that is not up to date. A new version will appear soon.

1. Brief introduction to the SJA

This document summarizes the discussions at the Webyog Forums about the use of the Synchronization features of the SQLyog Job Agent (hereafter "the SJA" or just "SJA") provided with the enterprise version of Webyog's flagship product SQLyog Enterprise and also available as a free utility for the LINUX operating system.

The sync tool is an easy-to-use tool to bring MySQL databases "in sync". In has options to make it equally well fit for use by beginners as well as expert users. In situations where MySQL replication is not possible for some reason – such as that you don't have the privileges for that or you share one or both servers with other users – the SQLyog sync tool will do the job for you.

I am far from being a database expert, but I have been following the discussion at the Webyog Forums regularly and have been using the tool quite a lot myself too. In this document, I will address some of the common misunderstandings about the SJA and give some hints on how it can be used effectively. Some of those hints are resulting from my own experiments and observations, for others I must give my thanks to the other people at the Webyog Forums – too numerous to mention. The data sync facility of SJA is now about two years old. There have been a few minor changes, bug fixes and performance improvements and recently one major enhancement (sync without a Primary Key), but in most respects it is the same program as it was from the very beginning.

With this paper, I also hope to raise a discussion on how it should develop in the future.

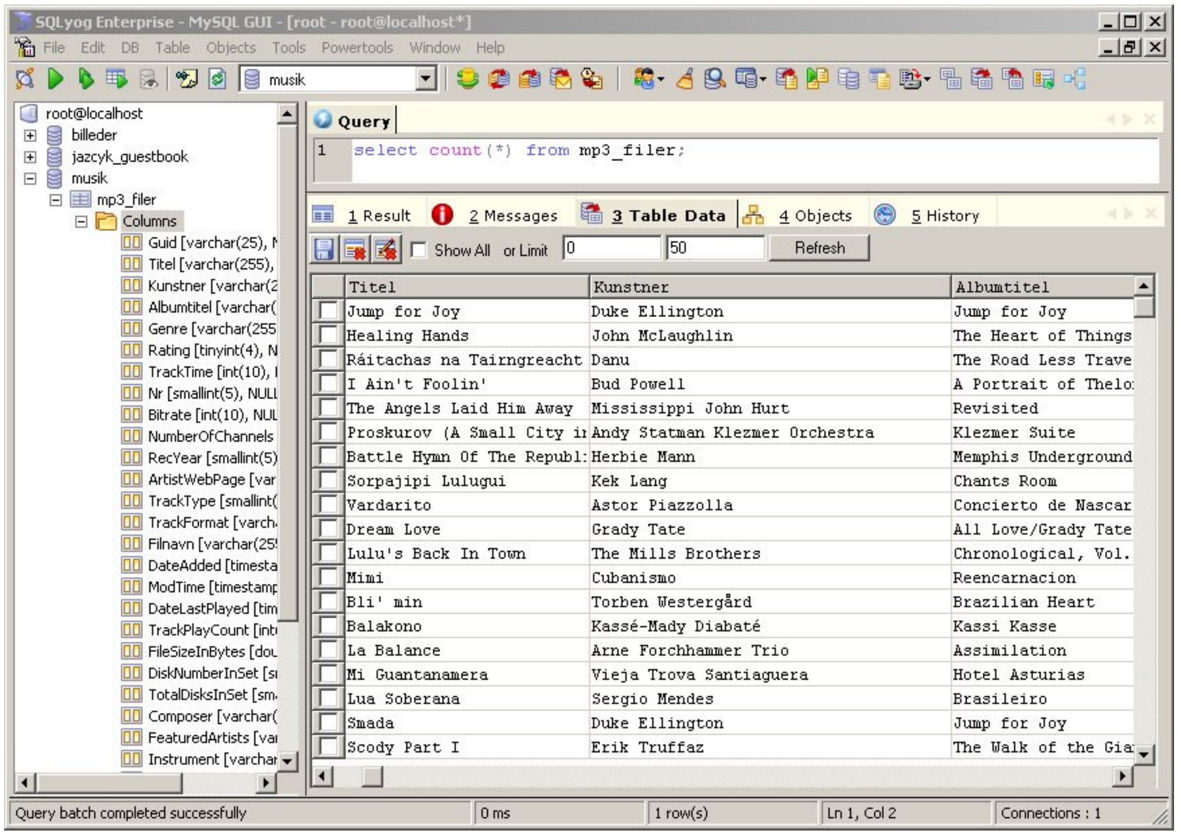
This is the forth write-up of this document. With second write-up first and foremost the (rather important!) paragraph "combining things" was added. The primary purpose with the 3rd write-up was restructuring the article: primarily to "put the beginners' stuff at the beginning" to make it easier for not-so-experienced users to get started. And finally this fourth write-up deals with sync without a PK and adds a few considerations on the use of the sync tool with partitioned tables as of MySQL version 5.1 that is being released for the first time as official MySQL binary distributions right now. And the graphics has been replaced with graphics showing the wizard as of SQLyog version 5. And a few discussions at the Webyog Forums have given cause for some additional remarks.

The content of this article will be:

- ✓ Some common situations where SJA can “save the day” for you
  - ✓ What to be beware of.
- 5. **Inefficiencies of web hosting** and workarounds with the SJA
  - ✓ My own story
- 6. **Combining things**.
  - ✓ Using more options at the same time
  - ✓ Combining the sync-feature of SJA with other features of SJA
  - ✓ Return to music database example
- 7. **Some other few simple tricks** on using the SJA.
  - ✓ Choose an appropriate Primary Key
  - ✓ Making the Primary Key host-specific
  - ✓ Run multiple instances of SJA at the same time
  - ✓ Running SJA in an infinite loop from a batch file
- 8. **Sync’ing tables without a Primary Key**.
- 9. **Some wishes for the future** development of the SJA

2. Getting Started. The easy approach to using the SJA for sync.

The example: the music database



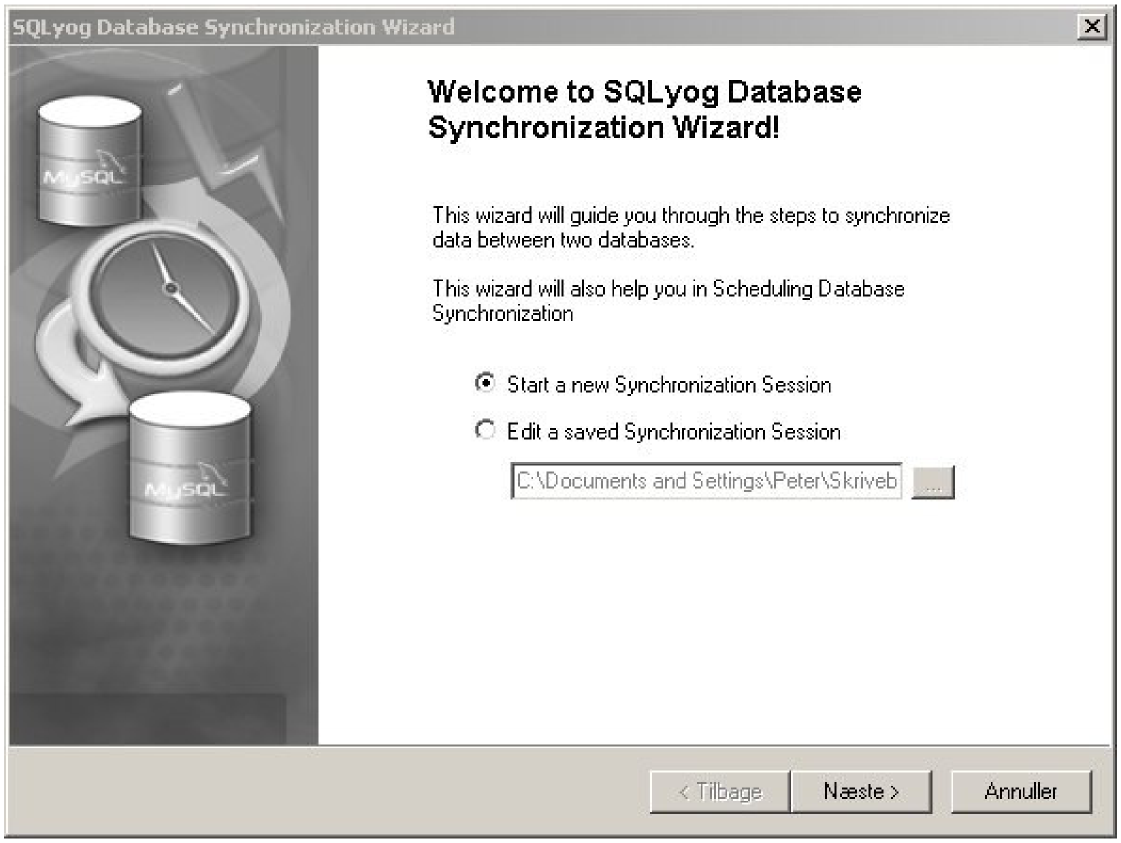
```
select count(*) from mp3_filer;
```

Titel	Kunstner	Albumtitel
Jump for Joy	Duke Ellington	Jump for Joy
Healing Hands	John McLaughlin	The Heart of Things
Ráitachas na Tairngreacht	Danu	The Road Less Trave
I Ain't Foolin'	Bud Powell	A Portrait of Thelo
The Angels Laid Him Away	Mississippi John Hurt	Revisited
Proskurov (A Small City ii	Andy Statman Klezmer Orchestra	Klezmer Suite
Battle Hymn Of The Republ	Herbie Mann	Memphis Underground
Sorpajipi Lulugui	Kek Lang	Chants Room
Vardarito	Astor Piazzolla	Concierto de Nascar
Dream Love	Grady Tate	All Love/Grady Tate
Lulu's Back In Town	The Mills Brothers	Chronological, Vol.
Mimi	Cubanismo	Reencarnacion
Bli' min	Torben Westergård	Brazilian Heart
Balakono	Kassé-Mady Diabaté	Kassi Kasse
La Balance	Arne Forchhammer Trio	Assimilation
Mi Guantanamera	Vieja Trova Santiaguera	Hotel Asturias
Lua Soberana	Sergio Mendes	Brasileiro
Smada	Duke Ellington	Jump for Joy
Scody Part I	Erik Truffaz	The Walk of the Gia

```
Query batch completed successfully | 0 ms | 1 row(s) | Ln 1, Col 2 | Connections : 1
```

This is my (Peter’s) own personal music database as it looks in SQLyog version 5.0. I preferably use the MusicMatch Jukebox as my Music Player. The most recent version of this player builds its "Music Library" on an Access database. These data that you see here is a ("vertical") subset of the data from that Music Library that are exported from that Access database into MySQL with the SQLyog Migration Tool. With "vertical" I mean to say that all rows are here but not all columns. I have a copy of the MySQL database at my localhost and a copy at my webhost. From time to time new songs are added, some data may be manually entered or corrected from the keyboard, Music Player itself updates some columns with data it finds on the internet, and updates time for latest playback, total no.s of playbacks etc. So after some time I would like to update the database at my webhost with the latest data from localhost. This is easily done with the sync tool of the SJA. The column "filnavn" (Danish for "file name") is the Primary Key of both the tables. There can't be two identical filenames in a file system, so each row of data is identified uniquely by the filename.

So let's start the sync tool GUI and see what it looks like. You start it from the "Powertools" menu in SQLyog. First dialogue looks like this:



SQLyog Database Synchronization Wizard

Welcome to SQLyog Database Synchronization Wizard!

This wizard will guide you through the steps to synchronize data between two databases.

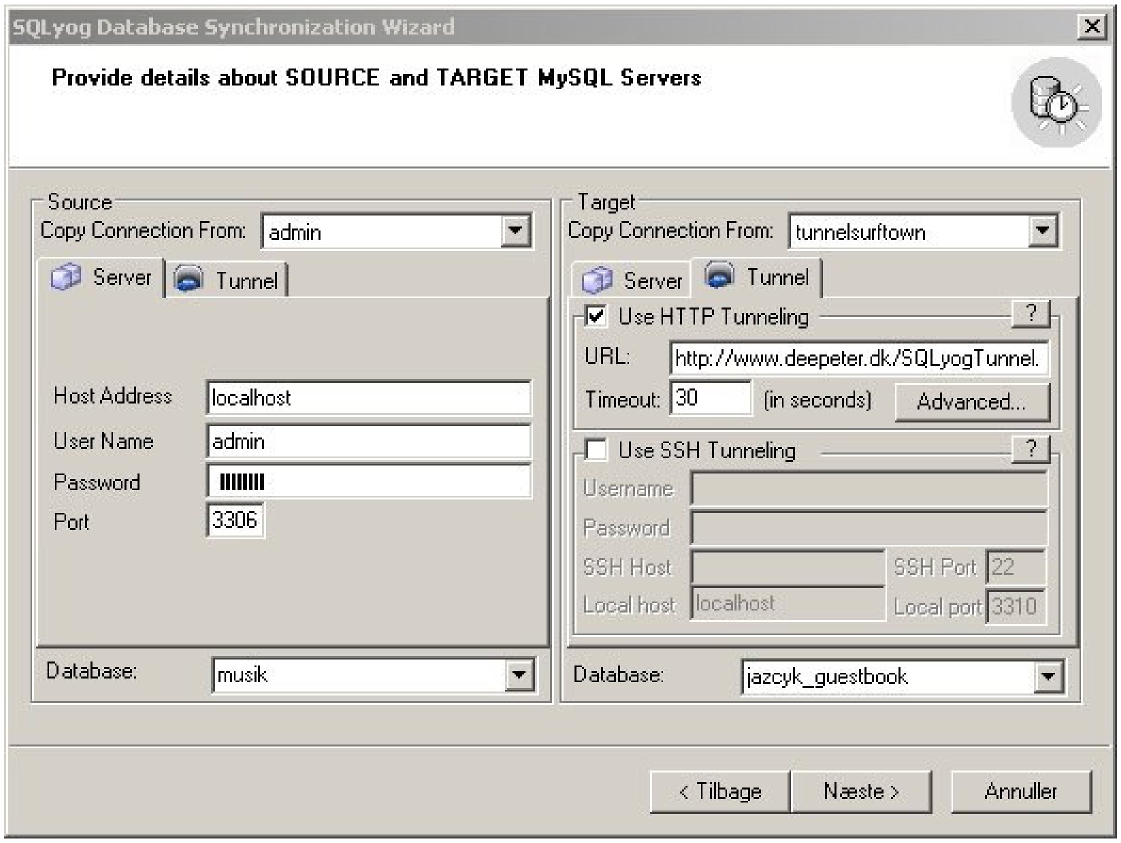
This wizard will also help you in Scheduling Database Synchronization

(●) Start a new Synchronization Session

( ) Edit a saved Synchronization Session

C:\Documents and Settings\Peter\Skriveb

Since we never used the sync tool before, we will start a new Synchronisation Session, and click "Next". Doing so brings this dialogue – the connection dialogue. Here I must enter the details necessary to logon to the two MySQL servers involved. The Server tab contains the information that the MySQL server needs, the Tunnel tab the information that the networking environment needs to let the "let the request to MySQL". Refer to SQLyog help file for details on that. You can note here that I use HTTP-tunnelling with my webhost. It is the only option I have. Direct connection is blocked and SSH-tunnelling is not available.


SQLyog Database Synchronization Wizard

Provide details about SOURCE and TARGET MySQL Servers

Source

Copy Connection From: admin

Server | Tunnel

Host Address: localhost

User Name: admin

Password: ||||||||

Port: 3306

Database: musik

Target

Copy Connection From: tunnelsurftown

Server | Tunnel

[✓] Use HTTP Tunneling

URL: http://www.deepeter.dk/SQLyogTunnel.

Timeout: 30 (in seconds)

[ ] Use SSH Tunneling

Username:

Password:

SSH Host:

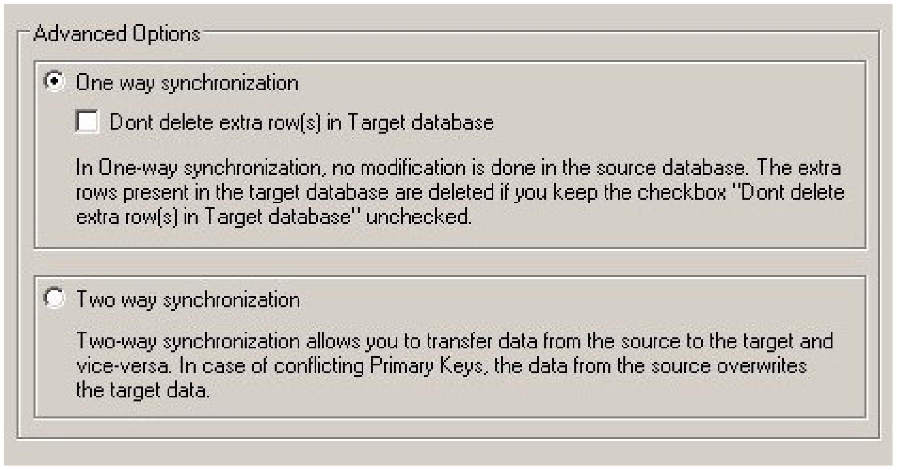
SSH Port: 22

Local host: localhost

Local port: 3310

Database: jazcyk_guestbook

In this case the database at the webhost is 'static' between sync's. There has not been any change there since last sync – on the contrary it is the changes at the localhost that should be effectuated at the webhost. That's why I choose "One-way synchronisation"


Advanced Options

(●) One way synchronization

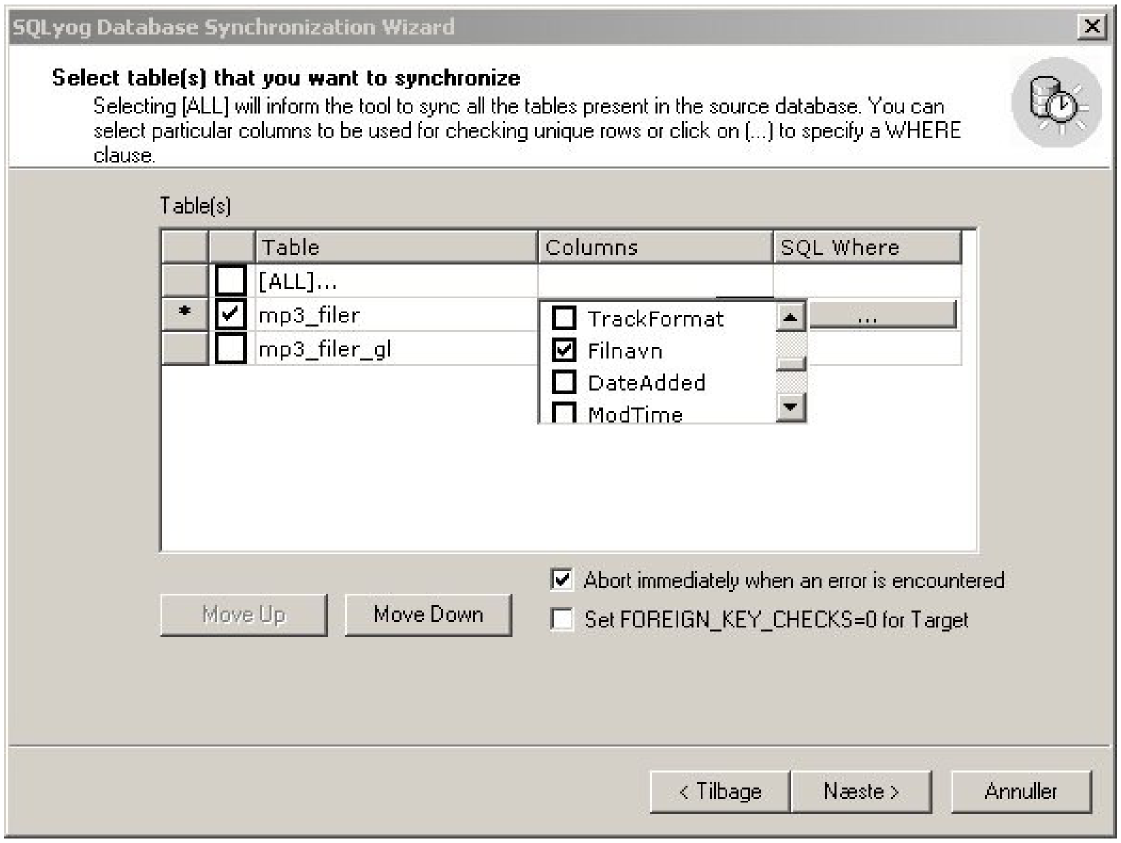
[ ] Dont delete extra row(s) in Target database

In One-way synchronization, no modification is done in the source database. The extra rows present in the target database are deleted if you keep the checkbox "Dont delete extra row(s) in Target database" unchecked.

( ) Two way synchronization

Two-way synchronization allows you to transfer data from the source to the target and vice-versa. In case of conflicting Primary Keys, the data from the source overwrites the target data.

Next dialogue is where I choose what tables to sync. At the localhost I have the most recent result of the import from MS Access and the one before that ("gl" is short for "old" in Danish). Only the first (the most recent data) are sync'ed. We will deal we the other options available later.


SQLyog Database Synchronization Wizard

Select table(s) that you want to synchronize

Selecting [ALL] will inform the tool to sync all the tables present in the source database. You can select particular columns to be used for checking unique rows or click on (...) to specify a WHERE clause.

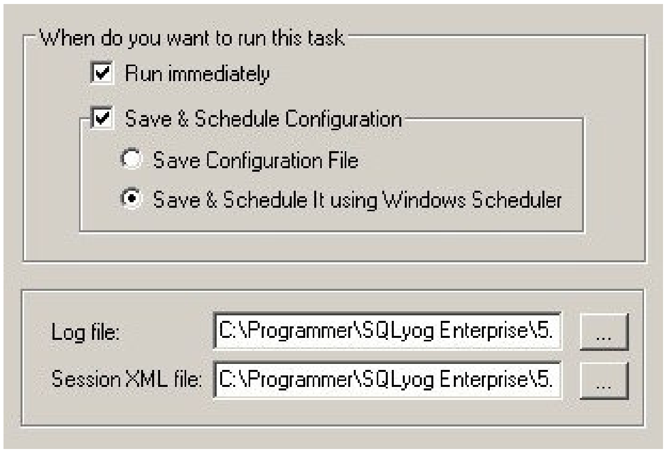
Table(s)

		Table	Columns	SQL Where
	☐	[ALL]...		
*	☑	mp3_filer	☐ TrackFormat, ☑ Filnavn, ☐ DateAdded, ☐ ModTime	...
	☐	mp3_filer_gl		

☑ Abort immediately when an error is encountered

☐ Set FOREIGN_KEY_CHECKS=0 for Target

That's all. We are ready to perform the sync now. We can start it immediately, schedule it or just save the description of the sync-job for later use.


When do you want to run this task

☑ Run immediately

☑ Save & Schedule Configuration

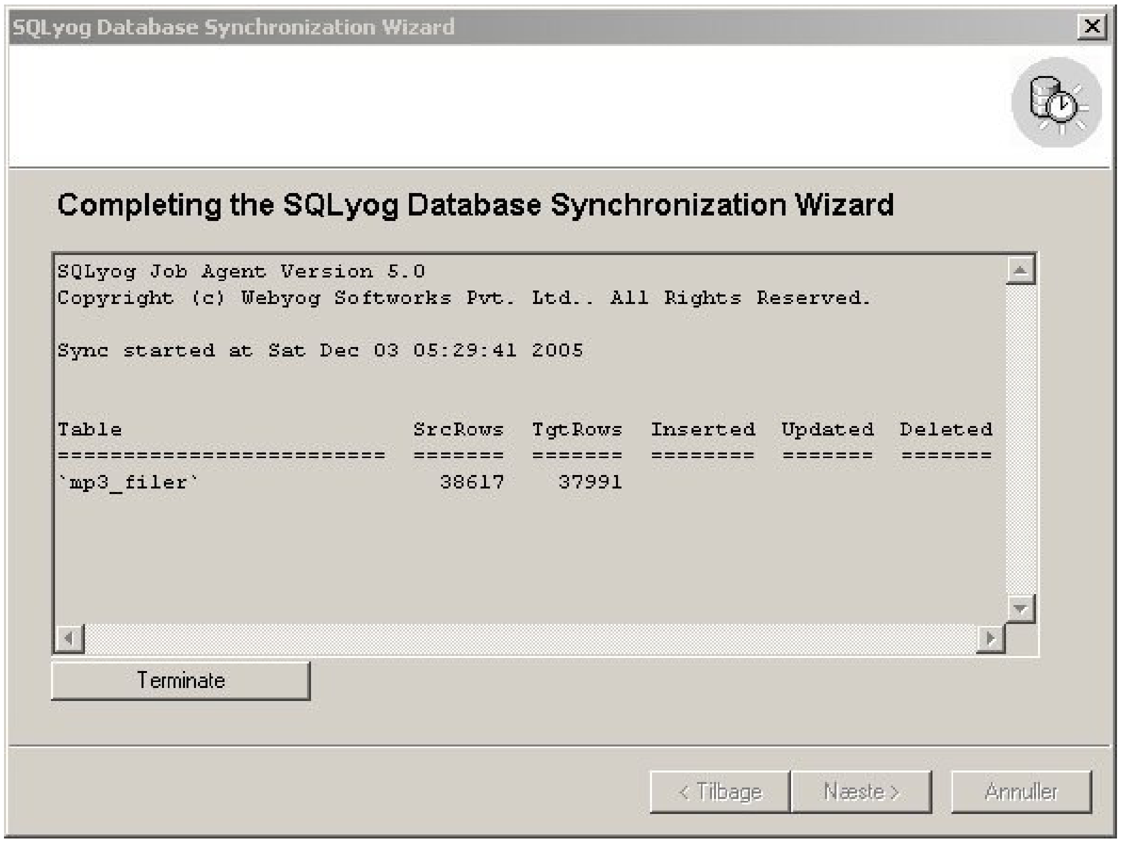
○ Save Configuration File

● Save & Schedule It using Windows Scheduler

Log file: C:\Programmer\SQLyog Enterprise\5.

Session XML file: C:\Programmer\SQLyog Enterprise\5.

Now the job has started.


SQLyog Database Synchronization Wizard
Completing the SQLyog Database Synchronization Wizard
Terminate
< Tilbage
Næste >
Annuller

```
SQLyog Job Agent Version 5.0
Copyright (c) Webyog Softworks Pvt. Ltd.. All Rights Reserved.

Sync started at Sat Dec 03 05:29:41 2005


Table                         SrcRows  TgtRows  Inserted  Updated  Deleted
=========================     =======  =======  ========  =======  =======
`mp3_filer`                     38617    37991
```

If the conditions needed for the synchronization process are met (most important: if tables are identical in structure), the sync tool starts, compares data at the two hosts and finishes after a while. At last the number of rows inserted, updated and deleted are displayed.

3. About the SJA and the sync tool more in detail.

What SJA is and what it does

The SJA is a standalone command-line MySQL client available for WINDOWS and LINUX (the Windows version, however, can be started from inside the SQLyog GUI as well as we have already seen). To put it clear: the SJA will run on Windows and Linux, but will connect to any MySQL servers running on any OS. That is the meaning of the term “MySQL client”. One of the most important features of the SJA program is to “synchronize” data (complete databases or individual tables) on two servers/host. It utilizes advanced checksum algorithms for comparison of the data and can be very effective in “finding out” which data must be INSERTed, UPDATEd or DELETEd on either of the hosts to bring them “in sync”.

Not only is the sync-functionality ‘one of the most important features of the SJA‘ it probably also is – to judge from the number of questions asked at the Webyog Forums - the most popular and requested one. It is also continuously improved with new features and performance improvements. Recent improvements (with SQLyog version 5.0) include an improved wizard for controlling it, an option to sync tables without a Primary Key. Further improvements are planned for SQLyog version 5.1 planned for release in the first half of 1st quarter of 2006.

These improvements include performance enhancements and an integrated e-mail functionality similar to the one that you find in the SQLyog Migration toolkit, if you should happen to know this one.

There are different options available with the sync tool: most important there is “one way sync” or “two way sync” available. Another important option is that you can choose never to let the SJA delete data. You can choose to let the program abort in case of some error or you can let it go on with the next row of data. There are some optimization options too, that will be dealt with later in this document. Finally, there is a FK(Foreign Key)-CHECK option that lets you use SJA for sync even if job should fail with a Foreign Key constraint check.

These options are read by the SJA from an easily understandable XML-file. SQLyog Enterprise provides a GUI to generate and edit this XML-file, but it – of course – might as well be entered from the keyboard or written by another application. SJA can also be launched by any application able of launching external programs and such program may even generate or modify the XML-file itself. That is one of the basic ideas of using the XML-format!

SJA may be installed on one of the computers running either of the MySQL servers and thus make one connection to “localhost” and another to some “remote host” (that probably is how it is mostly used!), but nevertheless, it is still a client acting as such in relation to both MySQL servers, just as it is when it is connecting to two “remote hosts” and runs from a third computer. One important implication of this is that all the data that the SJA must use, must be transferred from the servers to the machine and the memory area that SJA uses. Although I knew that in principle when I first tried the SJA, it gave me some surprise (that I will describe later).

One more thing about the SJA that you might appreciate is that all data are available for other users all the time during the sync. That might make you decide to use the SJA even in situations where importing a DUMP from one server to another could be used as well.

The Windows version of SJA takes advantage of the advanced connectivity features of SQLyog Enterprise: different tunnelling options (HTTP- HTTPS- and SSH-tunnelling) and the ability to work from behind a proxy. The different tunnelling option allows you to connect to your MySQL database(s) even if your ISP/Admin disallows remote connections

The basics.

I shall here make a few comments on some very important and basic aspects of using the SJA. If you think that all of it is obvious, I can guarantee, that it has not been so to everyone over the about 2 years the SJA sync tool has been in existence!

1) The first basic thing you MUST understand about the sync functionality of the SJA is that it will always need to have one of the hosts defined as “source” and the other as “target”. That’s also true for two-way sync. We will see the implications of that later.

2) The next basic thing you MUST understand about the SJA is that it will need to uniquely identify each row of data so that corresponding rows on both hosts can be identified. SJA is – as any computer program would - only able to use the information available with the data themselves for this. Remember that any information that you may have in your head (because it is your data and you know their meaning and their history!) is not available for SJA.

It is generally recommended to use a Primary Key (PK) of the table(s) to be synced for this unique identification. When a PK exists, SJA simply identifies each row of data by the PK. Two rows of data on either host having the same PK will be considered as identical – or in case they are not – as should-be- made-identical.

When a PK is not available each row is uniquely identified by the **complete** dataset (all columns) of the row. You may think of it as if a 'virtual key' being built from all columns is used. It is not exactly how the code works, and we will go a little deeper into it later.

3) It is also very important that you understand that in case of "conflicting PK's" (that is that same value of the PK is used by different data at each host), then data from the "source" will overwrite data at the "target". So the importance of setting up the PK with care can't be emphasized enough if you want to use the SJA with your data.

There might be special situations where you don't want to have a PK at all (or temporarily drop it) to avoid this situation and perform the sync and keep both rows of data duplicated on both serveers. It is now possible (from SQLyog version 5). But nothing comes for free. There are some other drawbacks with this. We will discuss it somewhat more in detail at the end of this article.

The situation when "conflicting PK's" has resulted in deletion of data, have given cause to some discussion over the years.

An excurse on the word "synchronize". Some discussions have taken place over what is actually the meaning of the word. The original word "synkron" I believe is Greek and REFERS ONLY TO TIME. It simply means "at the same time". So it's not hard to imagine what it means to "synchronize two watches". With elementary knowledge of physics, you could also easily expand the meaning of the word from relating to TIME to relating to SPEED and it will still be very precise. But synchronization of DATA – what is it actually? There is no authoritative answer (Yes – I have been checking quite a few quality dictionaries in different languages!). Just let us conclude here that the result of synchronization is IDENTITY of the data. But there might more than one result of the synchronization process that could result in IDENTITY, and no one is more "correct" than the other as far as the meaning of the word goes. There has been some "fuzz" with the meaning of the word at the Webyog Forums. Some users expected that "synchronization" of data would never result in data being deleted. That was what the word meant to them. The only answer I can give to this is: before using a tool like SJA with real and important data, you should understand how it works!

Should we synchronize our beers? Cheers!

4) I'll also add here that SJA does not automatically use TIMESTAMPS or any time- or date-variable type (even if they are available) for sync unless the PK is built with such variable. I write this explicitly here, because that has also surprised some users. However the 'timeslice' or 'accuracy' of a TIMESTAMP with MySQL is (only) one second. And one second can be a very long time on fast servers performing a lot of operations. There will normally be several – maybe hundreds – of rows with the same TIMESTAMP –value on heavily loaded servers. So a TIMESTAMP alone will generally not be useable as a unique identification!

Finally I'll add that there is more introductory stuff to read on this in the Webyog FAQ. Do it if you like! And in the Webyog Forums questions and discussions regarding various aspects of using the sync tool (and the SJA in general) occur frequently, so you also could keep an eye here. There are both very simple questions and complicated questions being asked from time to time.

Also note that SQLyog Enterprise includes a **Structure Synchronization Tool** too. This tool is not implemented in the code of SJA but the 'core' of the SQLyog program itself. However they still can be used together in various ways.

What SJA is not and what it does not

The SJA is NOT a replication feature comparable to the replication feature of the MySQL server itself! That is probably the most common misunderstanding about the SJA. It is not either AT ALL any sort of "daemon" running with the MySQL server "supervising" or "over viewing" what the server is doing. As any MySQL client, it starts (when you or some application or some scheduling feature starts it), it does its job, it finishes, and it closes down when it is done! Another way to put it: The SJA does not access the MySQL server itself as such or its log files, but it accesses the DATA only that the server lets the client (in this case: the SJA) access.

More precisely and more technically: MySQL replication and SQLyog Synchronization are technically completely different. MySQL server replication basically operates in the way that the server defined as the 'slave' reads the binary log of the 'master' and replicates operations performed by the master. SQLyog/SJA synchronization uses SQL to query the servers, compares the data using advanced checksum algorithms and performs the INSERTS, UPDATES and DELETES needed to bring databases in sync. Both methods have their advantages and drawbacks. The most important aspect in favour of the method used by SQLyog synchronization is, that it is possible even if one or both of the databases are in a "shared environment" such as an ISP. You don't need any special privileges to use SQLyog synchronization - only simple SELECT, INSERT, UPDATE and DELETE privileges. And you won't need your own dedicated hardware. But of course SQLyog Synchronization can't compete with MySQL replication in terms of 'real-time performance'. That never was the idea either. Nevertheless we have users running a sync job each 5 minutes or even each minute. That can be done (depending on the amount of data and the frequency of changes in the data) with appropriate server configuration and reasonable bandwidth available. And – to our benefit - the more frequently sync is done, the less data must be written with each sync. And also it does no harm if one sync job occasionally starts before the previous has completely finished. It will put a little more load on the server, and that is all there is about it!

Also note that the sync tool is not a tool for SELECTion of data. The purpose of the program is to "synchronize" COMPLETE table(s) – not a subset of data from a table. With the SQL_WHERE option (that we deal with later) you might be able to generate a subset of a table, but this is not purpose of this feature. The SQL_WHERE option is an optimization option, not a selection feature!

4 .Why use the SJA?

Some realities about web hosting and Internet connectivity.

Some readers now might want to ask: Why then not just use the replication feature of the MySQL server itself? Why invent something new that seems not to "do as good a job as the original thing"? Well – basically there can be two reasons for that:

1) It may not be desirable
2) It may not be practicable

The situations where it is not desirable are all sorts of situations where you don’t want data to change without you asking for it yourself. That is particularly true in development situations and trouble-shooting situations. If the data you are working with change “behind your back” and without your knowledge, you can really be messed up. It just did this-or-that before, so why not now? So even if you do have the option to use MySQL replication, there might be situations where SQLyog synchronization is a better choice. Actually they can supplement one another.

The situations where it is not practicable can be more complex, but very often it comes back to the simple fact (that we have discussed) that in order to use the replication feature of the MySQL server itself, you must have at least two MySQL servers running that are completely under your control. And further they must have connections (bandwidth, reliability) appropriate for what they are being used for. And finally, there should be someone watching the servers to take appropriate action if something goes wrong (could be anything from a server crash, some “deadlocks” or even any sort of attack/hacking attempt from the internet). That’s not a realistic situation for most individuals, small companies, and even not for a lot of medium-size companies. For practical reasons, you will often choose web hosting at some hosting provider, and rely the professionalism available here. But then, you typically will share the database server with dozens, hundreds or maybe even thousands of other users, and most often you will have to agree to some not very flexible conditions. You can take it or leave it. That of course applies to web-based solutions rather than traditional corporate intranet-based solutions (where for instance bandwidth is a much more simple issue and where threats and attacks can be much more easily handled).

Of course, web hosting differs in quality (and price!), but most often and in practice, the use of replication is out of question. But then, there is a good chance that the SJA will do the job for you. It may take some considerations, some planning, might involve some restrictions on what’s allowed to do with the data, and even in some cases, some additional coding with your applications. But I believe that in most cases, you are almost done when you have understood the way the SJA works.

Some common situations where SJA can “save the day” for you.

- You are a developer and want to test your code with the most recent data from the production server. SJA will bring the production DB and the development DB in sync.
- Some of your corporate data must be available to the public or your partners over the Internet. However – for security reasons or any other reason – you don’t want those people to access your production server, and will have to set up another server (could be with a hosting provider for instance) with a copy of some of your data. With SJA, you can sync data at regular and scheduled times.
- You are doing business in a part of the world where permanent Internet connectivity is not realistic. Setting up your IT-environment in a way that depends on working connections to a database server far away could result in a lot of unproductive hours. You may then choose to “double” your DB’s (or part of them) and use SJA when connection is available to bring data in sync.
- Service personnel, sales representatives, and other “travelling people” might want to bring with them a copy of certain data from the corporate server on a laptop. It could be for instance, spare-part lists, service instructions or financial information concerning the customers/vendors that you will come in contact with – anything. There might be an Internet connection at your hotel, but it might not work, so bringing the data with you could prove a good idea! And a service engineer doing

maintenance on some mining machinery “in the middle of nowhere” or 400 feet above the ground with a Wind Power Generator can not be sure to be able to reach the spare parts list or the service manual on the server of his company. It’s as uncomplicated to run a MySQL server on a laptop as on any PC hardware and with SJA the relevant and most recent data can be sync’ed before departure. That would ensure much more updated data than any printed manual.

- You’re doing a lot of travel and want to use the time spent with travelling (and waiting – since waiting seems to be an ever-rising by-product of travel!) working with your data. You can work with a copy of the database and with SJA you can sync the changes back to the corporate server or the server at the Internet hosting when arriving back at home.
- SJA can also be used as a supplement to traditional backup procedures.

What to be aware of:

There are always dangers and problems when two “copies” of the same data exist apart from one another and might develop independently. SJA can’t undo this reality – it is no magician! No matter what you do, it takes considerations on how to structure the databases, discipline when working with data, and might also take some extra code to “take hand” of situations that might occur due to the fact that the “master” server is not available. It’s your own responsibility as a user/administrator of SJA to structure data, define procedures for working with them, etc. I will address this in the next chapter of this document with some examples. However in cases where all changes (INSERTs, UPDATES and DELETES) are done on the one database and later on sync’ed to another server as in our introductory example and similar simple cases, it of course is quite uncomplicated.

5. Inefficiencies of web hosting and workarounds with the SJA

My own story

My first experience with the SJA started quite promising. I used the SQLyog GUI to create the XML-file, opened it in an editor just to see what it looked like. It looked right! Simple and easy to understand! I started the job, and I could see connections being established. The SJA informed me about the number of rows found at each host. That also looked very reasonable. And then: nothing happened! After some cups of coffee, still nothing had happened. I aborted the process and tried once more. Same result. Checking the Windows task-manager showed the process was running, but consuming very little resources. I then started a network monitoring program. I could see some traffic going on over my Internet connection. Transmission speed: approx 20 kbit/sec what is 10% of my bandwidth in outgoing direction (I have an asymmetric DSL-line). After a couple of hours, I aborted the job again. Started it once more just before I went to bed. Job was still running when I got up in the morning. Now I became sturdy! I’d let it run until it’s finished even if it should happen to live longer than I! It finished a couple of hours later. 11 hours for syncing a table of about 10 MB, where I expected that somewhat in between 500 - 1000 rows out of about 35,000 totally were not identical on the two servers. That was not at all was I had expected. But the result of the sync was perfectly as I would have expected it to be. But obviously, some research on the reasons for this not quite satisfying performance was needed!

1)
First of all, bandwidth was obviously not an issue here. There was plenty of it. But it was not used! But undoubtedly, the various buffer settings with the MySQL-server at my web hosting (and maybe also the PHP-configuration since I use HTTP-tunnelling) are very restrictive and "conservative". And of course that makes some sense too. There are thousands of users on this MySQL server – mostly individuals and small business users. Some of those may have dozens or even hundreds of visitors simultaneously connecting to the database – or rather trying to connect to the database. In order to "let everybody eat something", the server resources for each individual connection must be set low and visitors are only allowed small 'timeslices' of access to the server. That makes some sense when users are addressing the MySQL server from some .php or .asp webpage where often only a single or a few rows of data need to be read to 'populate' a web page with data. But when you are logged on as an admin to your own DB to perform some intensive maintanaince, it just does not "feel right". But I have no option to change it (and the web hosting is in the absolute cheaper end of the scale and there is absolutely no support available for an issue like this). I can take it or leave it. Or find another place where I probably will have to pay somewhat more.

2)
I found out that there was a problem with an application doing UPDATEs when there was no reason to do so (no change of data) resulting in a TIMESTAMP-field being automatically updated by the server. As a result of this (and of my own lack of understanding of how to configure the sync tool), each sync involved syncing quite lot of data that there absolutely was no need to sync – if it had not been for that changed TIMESTAMP-field.

3)
The database contains a lot of VARCHAR(255)s and some MEDIUMTEXTs of which most were empty or just contained a few characters. Only in a very few cases, the variables are "filled up". But SJA can't know and will have to examine all the data. And the servers will have to read them and send them to the SJA.

4)
The good question no. 1 is then: Will SJA **REALLY** have to compare all columns of data to decide which rows should be sync'ed. With my knowledge of the applications, using the data I know that if any change takes place with a row (no matter what change), I could list a few (five I think and none of them BLOBs or TEXTs) columns of which at least one would change. Well – then there really is no need for SJA to compare anything more than those 5 columns. And then I found that SJA actually had that option to specify certain columns to be compared, it's the <columns all=no> option in the XML-file. I had saved a copy of both tables before my first sync the day before so I repeated. WOW! Now it was "only" a little more than 1 hour. In the first run, it would have disappointed me too, but now was something different – after all it was about 8-10 times faster than first time!

5)
The problem with the TIMESTAMP-field I'd had and have already mentioned, gave me the idea to optimize even further. When I learned about the <columns all=...> option, I also discovered the <SQL_WHERE> option. I sync this remote DB regularly once or twice a week, so I know for sure that the only data that need to be sync'ed are either new data or data that have been updated or deleted within the last 7 days. So when defining a new TIMESTAMP-field named "modtime" with all the tables involved with the sync job like

```
`modtime` timestamp NOT NULL default CURRENT_TIMESTAMP on update CURRENT_TIMESTAMP
```

the MySQL-server will take care of updating the field automatically with **CURRENT_TIMESTAMP** when any other data in the row changes. Thus, there was even no need for a change in the code of any application. With SJA, I now can use the SQL_WHERE condition

```
<sqlwhere>modtime > current_timestamp - 100000000</sqlwhere>
```

or in plain words "do only consider rows not more than 10 days old or rows updated or deleted within the last 10 days". I'm now down at 8-10 minutes with a typical sync job (that typically involves comparing about 2000 rows and doing some operation on a little more than 500), and something like that was what I expected at the very beginning. There is one nag: it won't DELETE anything at the target even if it has been deleted at the source in case it has created more than ten days ago. This is due to the fact that the SQL_WHERE condition is tested with data on both hosts. With a one-way-sync, I think I should only test on the source. At least, that would be the "best fit" for my needs. But that is not the way it is as of now – and by the way, others may have other needs. However, that's not big deal in my case – I can repeat the sync without an SQL_WHERE in the relatively few cases where data have been deleted (yes this is a DB that constantly grows and rarely shrinks!)

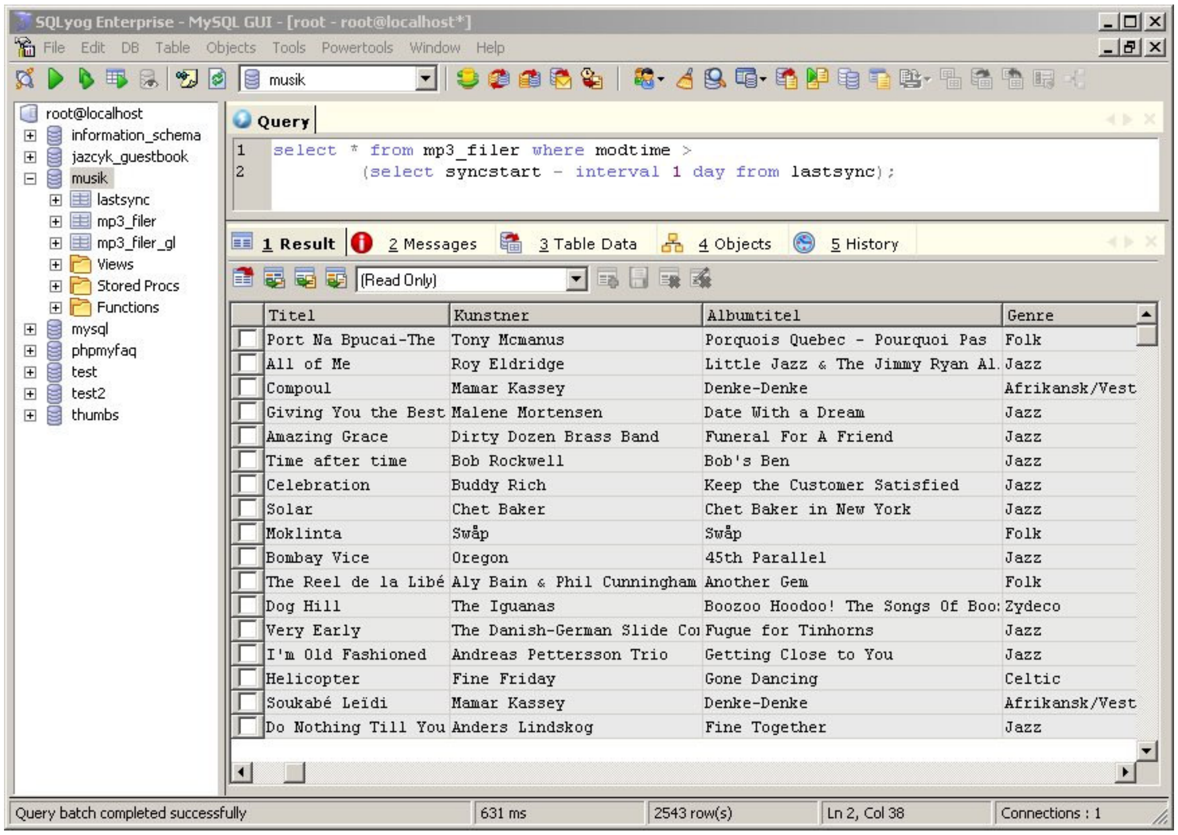
So my advice to you people who still hang on reading this: analyse your data and understand what the applications using these data do with them. You might then very well be able to use the <columns all=no> and <SQL_WHERE> -options in a way that dramatically increases the performance of the SJA by specifying a few columns for comparison or an appropriate WHERE condition. But YOU will have to analyze and describe it. SJA can't by itself.

I devoted quite some room here for describing how I ended up using this TIMESTAMP-field with the SQL_WHERE, simply because I think quite a lot of people might be able to use it in its original or some modified form.

With MySQL server version 4.1 or above, there is an even more elegant way of doing it using sub-queries. The images below demonstrate the principle. When sync is done (or started), you write CURRENT_TIMESTAMP to your database somewhere. With this example, a TABLE named "lastsync" is created to hold that single piece of data. You use that value with the SQL_WHERE in the XML-file (using sub-queries) with your next sync like

```
<sqlwhere>modtime > select syncstart – interval 1 day from lastsync</sqlwhere>
```

("Interval" could be "1 hour" instead of "1 day" as appropriate (a little 'overlapping time' will ensure that all that must be synced will be), and there could be both a "syncstart" and a "syncdone" value, and the table could be expanded to hold data from, for instance, the last 10 sync-jobs instead of only one.) Updating the LASTSYNC-table could all be done manually with some prepared SQL-statements or from within a simple application. It's all for you to work out your own solution!

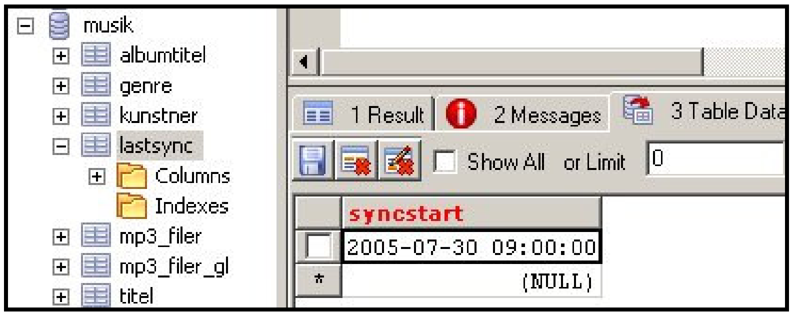


```
select * from mp3_filer where modtime >
        (select syncstart - interval 1 day from lastsync);
```

Titel	Kunstner	Albumtitel	Genre
Port Na Bpucai-The	Tony Mcmanus	Porquois Quebec - Pourquoi Pas	Folk
All of Me	Roy Eldridge	Little Jazz & The Jimmy Ryan Al.	Jazz
Compoul	Mamar Kassey	Denke-Denke	Afrikansk/Vest
Giving You the Best	Malene Mortensen	Date With a Dream	Jazz
Amazing Grace	Dirty Dozen Brass Band	Funeral For A Friend	Jazz
Time after time	Bob Rockwell	Bob's Ben	Jazz
Celebration	Buddy Rich	Keep the Customer Satisfied	Jazz
Solar	Chet Baker	Chet Baker in New York	Jazz
Moklinta	Swåp	Swåp	Folk
Bombay Vice	Oregon	45th Parallel	Jazz
The Reel de la Libé	Aly Bain & Phil Cunningham	Another Gem	Folk
Dog Hill	The Iguanas	Boozoo Hoodoo! The Songs Of Boo:	Zydeco
Very Early	The Danish-German Slide Co:	Fugue for Tinhorns	Jazz
I'm Old Fashioned	Andreas Pettersson Trio	Getting Close to You	Jazz
Helicopter	Fine Friday	Gone Dancing	Celtic
Soukabé Leïdi	Mamar Kassey	Denke-Denke	Afrikansk/Vest
Do Nothing Till You	Anders Lindskog	Fine Together	Jazz

```
Query batch completed successfully | 631 ms | 2543 row(s) | Ln 2, Col 38 | Connections : 1
```

Picture shows the use of subquery on music database with mySQL version 5 and SQLyog 5.0



syncstart
2005-07-30 09:00:00
(NULL)

The simplest setup: **lastsync** table with one value only – a timestamp variable holding the date and time of last sync.

6. Combining things.

1) Using more options at the same time

The first matter that we shall consider is how to use more different options (NEVER_DELETE, SQL_WHERE, COLUMNS_SELECTION) at the same time with the same job-file. That of course can be done, but you should take the time to understand how the different options "interact" or are "evaluated together". What I will describe here is my original understanding of it. It is **not** the way it is implemented in the code of SJA itself. For instance, if you have been using SJA for two-way sync already, you may have noticed that a two-way sync is executed as a one-way sync from source to target followed by another one-way sync from target to source. That is not how I will describe it here. But I believe that this description of mine fits better to human brain (as the SJA code itself fits better to a computer!) than any "linear" description following the structure of the SJA code itself.

Now, let's say you have a jobfile containing all these options: NEVER_DELETE, SQL_WHERE, COLUMNS_SELECTION. What will happen then when you execute it? I'll describe it as an evaluation process that consists of 3 steps. We will here consider sync'ing tbales with a PK:

1) **Evaluated first: SQL_WHERE:** SQL_WHERE selects rows that should be considered further in the process. SQL_WHERE is executed on both hosts (what I think it should not always with a one-way sync!) and any row on either of the hosts that fulfils the WHERE-condition (and ONLY those rows) proceeds to next step. If there is no SQL_WHERE specified, all rows proceed.

2) **Evaluated next:** PK's of the rows selected with step 1).
   - If a row selected with 1) has a PK-value that exists on source only, that row will be INSERTed on target.
   - If a row selected with 1) has a PK-value that exists on target only
     - **with a one-way sync:** that row will **either** be DELETEd on target **or** there is no action taken (depending on whether "NEVER_DELETE"-option is set)
     - **with a two-way sync:** that row will be INSERTed on source.
   - If a row selected with 1) has a PK-value that exists on both hosts that means: proceed to step 3.

3) Evaluated last: Data are compared.
With rows that have got as far as here (rows with identical PK's on both hosts), a string is built with the concat_ws() SQL-function using the columns selected with the COLUMNS_SELECTION as arguments, and checksums are calculated with that string on both hosts. If checksums are identical, data are considered identical too and nothing is done – if checksums are not identical, source data will overwrite target data for those columns where data are not identical. **Note that not only** columns selected with COLUMNS_SELECTION-option are overwritten but **all** columns of the row where data are not identical. The COLUMNS_SELECTION-option specifies which data should be compared only – not which should be sync'ed! The **complete** row (all columns) will always be sync'ed by SJA!

But don't forget to remember that this 3-step explanation is my "personal translation" of the workings of SJA. Code design/implementation/optimization is different!

Another important aspect to understand here is that the concatenation of the <columns>variables is performed on the server side, while generation of checksums and further calculations is performed on the client side. The understanding of this is important when trying to analyze what possibly could be done to increase the performance of the sync tool. If the client is overburdened with other tasks it could be the 'bottleneck' in the process. Or it could be the server (it's configuration and resources available for the connection) or simply insufficient bandwidth.

Unfortunately the implementation of the concat_ws() function has changed with various MySQL versions. If you try to sync across versions where this is the case, SJA reports an error stating that concat_ws is used and this might not work with different versions of MySQL - and aborts. There are no plans to 'get around this' by implementing the concatenation on the client side – simply because that would completely destroy the efficiency of the tool. Thus the sync tool cannot be used if the MySQL versions are "too much different" in this respect. Version 3.23.x cannot be sync'ed with 4.0.x and higher and 4.x cannot be synced with 5.x.

2) Combining the sync-feature of SJA with other features of SJA

Since the release of SQLyog 4.1, SJA accepts 4 different types of jobfiles, that we in some XML-like terminology may name as <backupjob>, <notifyjob>, <syncjob>, and (the new one with SQLyog 4.1) <importjob>. Each jobfile can only contain one job, but you can easily execute a sequence of SJA-instances using for instance the OS-facility for that (a .bat-file on Windows).

Let us have an example: You want to

1. **Import from an ODBC-source to a local MySQL-server**
2. **Some data need to be reformatted**
3. **You want to sync the newly imported and reformatted data with a remote MySQL server**
4. **Finally, you want to backup your remote database.**

The reformatting in step 2 could be simple things like:

- Removing trailing blanks from strings
- Converting UPPERCASEs to LOWERCASEs
- ROUNDing numerical values to fixed number of decimals

And even something more complicated (but not very!) like

- building a COMPLETENAME column from imported FIRSTNAME and SECONDNAME-columns and inserting data into that column using concat() or concat_ws() functions.

Now, a <notifyjob> could easily do what is needed for step 2. So to complete all four steps, you need only a .bat-file looking like:

```
sja my_importjob.xml
sja my_notifyjob.xml
sja my_syncjob.xml
sja my_backupjob.xml
```

SQLyog lets you build the appropriate XML-files from its GUI and the .bat-file itself can be scheduled with the Windows scheduler (or any other scheduler running as a service with the OS).

3) Return to music database example

If we return to the initial music database example we would then - if we wanted to execute import, reformatting and synchronisation from one bat-file. – need to have a .bat file like

```
sja musikimport.xml   //an importjob
sja musikrepair.xml        //a notifyjob
sja musiksync.xml     // a syncjob
```

In this case the SQL-statements in the <notifyjob> are

```
update track set filnavn = trim(trailing RIGHT(filnavn,1) from filnavn);
```

```
drop table `musik`.`mp3_filer_gl`;
rename table `musik`.`mp3_filer` to `musik`.`mp3_filer_gl`;
rename table `musik`.`track` to `musik`.`mp3_filer`;
```

First line is rather special! The reason for it is that there is a bug with the Music Player when it gathers information from the file system. It puts one end-of-string encoding character sequence (“\0”) too much in the end of the filename-variable. Even that can be corrected before data are synced to the remote server. And it needs to be done. Because it makes the concat_ws() function behave “strange” resulting in a failure to sync. Below is the total display in the command-line box when that series of job is executed as a .bat-file (“musikupdate.bat”) from command line.

```
C:\Programmer\SQLyog Enterprise\4.1>musikupdate

C:\Programmer\SQLyog Enterprise\4.1>sja musikimport.xml
SQLyog Job Agent Version 4.1
Copyright (c) Webyog Softworks Pvt. Ltd.. All Rights Reserved.

Job started at Sun Aug 28 05:11:39 2005

DBMS Information: ACCESS
Importing table schema: Track... Successful...
Importing table data: Track...
1000 rows transferred!
2000 rows transferred!
3000 rows transferred!
etc...
38000 rows transferred!
38617 rows transferred!
Successful...

Total time taken - 58 sec(s)


C:\Programmer\SQLyog Enterprise\4.1>sja reparermusik.xml
SQLyog Job Agent Version 4.1
Copyright (c) Webyog Softworks Pvt. Ltd.. All Rights Reserved.

Job started at Sun Aug 28 05:12:37 2005

Mail send successfully

Total time taken - 3 sec(s)


C:\Programmer\SQLyog Enterprise\4.1>sja musiksync.xml
SQLyog Job Agent Version 4.1
Copyright (c) Webyog Softworks Pvt. Ltd.. All Rights Reserved.

Sync started at Sun Aug 28 11:12:40 2005

Table        SrcRows  TgtRows  Inserted  Updated  Deleted
========     =======  =======  ========  =======  =======
`mp3_filer`    38617    37991       626      427        0

Total time taken - 356 sec(s)
```

The complete process of updating the remote host took in this case about 6 minutes - with appropriate <columns> and SQL_WHERE -settings of course.

And that is total for importing (quite a lot, actually) data about 90 GB of music files, correcting a defect with the data and sync'ing changes to a (slow!) remote server over the Internet.
With the different options available for the various <jobtypes>, it seems like only imagination limits your possibilities! And of course any other OS/system command or executable file can be included with the .bat-file.

7. Some other few simple tricks on using the SJA.

Choose an appropriate Primary Key

What is then an "appropriate Primary Key"? It's impossible to give an answer that's the right one in all situations. But I have noticed some people getting into problems when they use an auto-increment integer as a PK and at the same time letting both of the two databases expand between sync. To put it down to earth: if you have 46 rows in a database with an integer PK, the next row added will have a PK-value of 47. If you have two identical DB's with 46 identical rows and add a new row (with different content and meaning) to both, they will both have a PK-value of 47. With the next sync, you will have the "conflicting primary keys" and that row on the target will either be overwritten with data from the source or (in case you have set the "never delete data" option) they will not be sync'ed. Neither is what you want. Quite a lot of users have had the expectation that both rows of data would exist on both hosts after the sync. But that is not how SJA works! If that's your situation, check your data to see if there is another column or some combination of columns that would be unique and could be used more safely as a PK. I believe that an auto-increment integer PK is probably the worst solution for a PK if you use your DB's this way!

Making the Primary Key host-specific

If you want to be 100% sure that you don't lose newly created data with the sync and also want to ensure that all data is synced, you can build a PK composed of any field(s) that you would normally for a PK AND any other having different DEFAULT value on the two hosts. It doesn't matter if that variable is a number-type and has defaults for instance "0" and "1" respectively or it's a char-type having defaults for instance "webhost" and "localhost". With this new PK, "conflicting primary keys" will not occur during sync, since the same value for the PK would never exist on both hosts with data created since last sync. The host-specific part of such PK will always tell on which host the row was first created and it will work fine with SJA and INSERTs and UPDATEs (as long as there has only been an update on one of the hosts – but after all only updating data on one server between sync's must be elementary discipline necessary in situations like these). But think over then to how to be able to DELETE rows – the problem you run into here is that deleting from one host won't do if you run a two-way-sync afterwards! You will have to DELETE from both hosts between syncs, if not, the deleted row shall be written back from the host where it was not deleted. It is just an example of the "dangers" of having two DB's in principle containing the same data and letting them develop independently. It's your job as SJA-user or DBA to think the consequences of different situations to an end and take appropriate action, since each server will not be able to reach the data located on the other host.

Run multiple instances of SJA at the same time

When I (rarely) need to run a SJA sync-job that I can foresee would run for hours due to the conditions with the web hosting, I sometimes start more instances of SJA with the same parameters. Running about 5 or 6 instances simultaneously would make the whole process complete 3-4 times as fast as with one job only. I believe I "cheat" the restrictive settings at my web host this way. But I advise you not to use this trick too intensively on a corporate network if you still want to have friendly relations with your Sys Admin!

Running SJA in an infinite loop from a batch file

You can run a sync-job in an infinite loop with a batch-file looking like (Windows' version).

```
:label1
sja myjob.xml
goto label1
```

This will make a new sync-job start as soon as the old one terminates. I have been using that trick from time to time when the server at the web host is performing poor – something that happened periodically in a period. I experienced that in particularly some UPDATES were skipped under those circumstances. I then simply let it run repeatedly until it has displayed "0 inserts, 0 updates, 0 deletes" a couple of times and there are no error messages written to the logfile (of course the "abort_on_error" option must be set to "no" with this case).

8. Sync'ing tables without a Primary Key

The ability to synchronize tables without a Primary Key was introduced with SQLyog version 5. This feature was added due to so-called 'public demand'! There are basically two situations where users have requested this:

- In some cases applications will only accept database schemas that have a very specific structure - a structure generated by the application itself. Sometimes even adding a PK to the tables where it does not exist will cause the application to refuse working with the database. Even in such and similar cases you can now use the SQLyog Synchronization tool.
- Individuals running a database-driven website just for hobby, for small business or for some sort of organisation have argued that since they alone manage the data they can handle the potential risks of not having a Primary Key. They simply know the data that should be there. And the way the sync without a PK is implemented the risk of having data deleted in case of 'conflicting Primary Keys is eliminated. Typically this 'small-scale webmaster' himself performs some changes to the data (often using an laptop working disconnected) while at the same time the visitors to the website may change the same rows of data. With a PK that would result in 'conflicting Primary Keys' and one of the changes would be lost in the sync. However (as I also wrote at the beginning) when no PK is available each row is identified by the complete dataset in the row, and the two rows (that both have been changed) are now considered to be two different rows and the

sync would duplicate them on both servers. Thus neither the changes of the webmaster or the visitor is lost! However - it takes some manual control and possibly some maintenance scripts to avoid that the database get filled up with garbage since this also happens if the row is changed at one server only (and then probably the row should not be duplicated!). It meets the needs of some users this way – and they have wanted it!

Let us now describe more precisely how sync without a PK works:

1) First a number is retrieved by executing this SQL on the target with **data** being taken from the source:

select count(*) from targetdb.table where col1=data, col2=data.....

If the count(*)-number is not zero in the target then the row exists in the target (as well as the source). Then there is not anything more to do with that row (row exists in both source and target) and the same operation is repeated for the next row of the source.

2) If the count(*)-number is zero then the row does not exist in target and a:

insert into targetdb.table values (data1, data2,....)

is generated and executed it in the target. This is the explanation that the two rows (that both have been changed) are now considered to be two different rows. Actually there is no question of UPDATE as it is not possible know which columns(s) to update. The **select count(*)** statement only retruns a number (typically “0” or “1”). Or to put it another way: it returns information of the **existence** of the data – not about the data itself! And querying detailed information of each row of data is out of question for performance reason.

3) Finally SJA check for the need to perform DELETES: if the **count(*)** with **data** from the **target** returns a non-zero number when executed on the target and a zero-number on the source the row in target is deleted. Otherwise the process moves on to the next row.

Also note that the sync without a PK as it is described here is not able to distinguish between situations where there is one row with a certain dataset and more identical rows with the same dataset on the same server. More identical rows in source will result in sync to one row in target only and one row in source identical to more identical rows in target will not trigger any action. One row and more identical rows on the same host are simply considered identical situations.

If you don’t like it this way, then just don’t use it. Have a PK instead! Now you know what it is like. But without a PK we have considered this to be the optimal (taking performance into consideration). And (at least some) users have agreed!

9. Some wishes for the future development of the SJA

- I'd like a critical review of the XML generating GUI provided with SQLyog. Most (graphical) objects here are too small to my taste. "Real life data" could have very long identifiers/names and there could be dozens of columns with each table. When that is the case, the GUI is not optimal to work with as of now. However it **has** been improved quite a lot compared to the first version! And some users still use a screen resolution of 800*600. So maybe this will need to be a configurable option. With a 1600*1200 or higher resolution everything is very small on the screen!
- Documentation. Being a Command-Line utility, SJA will never be as intuitive as a GUI such as SQLyog. Although the XML-schema used by the SJA is not hard to understand, I feel that there is a need for a better documentation of the XML-schema – for the sync tool as well as for the other modules of the SJA. And preferably a 'formally correct' documentation as well as some almost trivial and easily understandable examples. However this has also been improved on over the last year or so.
- More logging options. As of now SJA displays a log message when it starts a sync-job, and thereafter only error messages and the concluding message. With sync-jobs that could be running for hours (without errors), I think this is somewhat too little. You simply don't know how far the job has proceeded or if it is "deadlocked" some way. Options could be an ability to display a message for each – say – 1000 records compared or for a time interval such as for instance 5 or 10 minutes. Also the error messages are rather sparse.
- An ability to use SJA with any unique index and not only a Primary key. Not so important now where sync without a PK is possible!
- More user-settings, especially whether the SQL_WHERE option should test on source only, target only or both. Actually the most important one to me!
- I have an idea that it would be useful to have an ability to use a symbolic addressing using "source" and "target" with the SJA as some sort of "reserved words" like "source.tablename.columnname" and "target.tablename.columnname" with the SQL_WHERE . That would allow for SQL_WHERE expressions like

  ```
  <sql_where>source.tablename.columnname >= target.tablename.columnname </sql_where>
  ```

  To avoid confusion: this construction should only compare rows from each table with identical PK's (or unique index if that was an available option)
- A simple "daemon" (written in PHP for instance) running in the server environment, but initiated and controlled by the SJA that would handle copy operations when copying data from one database to another on the very same server (with "the very same server" I more precisely mean all computers behind the same router/gateway sharing a global ip). If SJA could "snap in" to an existing PhpMyAdmin running at the server and use it that could be a solution too. As it is now, it seems somewhat meaningless that data must be transferred forth and back to the machine running the SJA when copying on one and the same remote host. But it is a logical consequence of the fact that SJA is a MySQL client. If the functionality of such a "daemon" could be expanded to an ability to connect to another remote server and thus allow for data transfer directly from one to another remote server that could be very fine too. The people at Webyog have demonstrated quite good programming skills in PHP with their HTTP-Tunnelling solution, so why not?
- MySQL ver. 5 related stuff: 1) **VIEW**s: it would be very nice if a VIEW on one host could be sync'ed with a TABLE (and a VIEW too, of course) on another. That would – I think – be a rather elegant addition that would also let you work with SJA and subsets of data in a safe way. 2) **TRIGGER**s, **STORED PROCEDURE**s and **FUNCTION**s:

Some consideration should be given how to handle these. Optimally, there should be an option to sync those as well – or at least display a warning if there are differences with TRIGGER and SP definitions of the two hosts. And two-way sync probably does not make sense here? What to do then? It must be considered and documented! And finally **PARTITIONED TABLES** (as of MySQL version 5.1): Well .. honestly .. I still don't know if this new feature of MySQL gives cause for some change in the SJA or if it opens up for new desirable opportunities. For certain partitioning models maybe syncing a partition on one server with a table (or a view) on another server maybe could be interesting – if it is possible at all. We will have to check up on that in near future!