Référence : SOU-PM-CRR- P-xxx

**P**

Soleil experience on Tango DB operation

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date de diffusion** | **Rédacteur** | **Vérificateur** | **Approbateur** | **Visa de l’approbateur** | **Modifications** | **Indice** |
|  | G.Abeillé | A.ButeauG.ViguierS.Le |  |  |  |  |
|  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |

Diffusion : Pour action / Pour information

Destinataires :

TITRE

Table des Matières

[1. Context 3](#_Toc420657460)

[2. Problems 3](#_Toc420657461)

[2.1. Data unconsistency 3](#_Toc420657462)

[2.2. Performances and monitoring 4](#_Toc420657463)

[2.3. Database backup 4](#_Toc420657464)

[3. Soleil thoughts for reflection 5](#_Toc420657465)

[3.1. Advices 5](#_Toc420657466)

[3.2. Data consistency 5](#_Toc420657467)

[3.3. Introducing ACID properties 5](#_Toc420657468)

[3.4. Audit and reporting 6](#_Toc420657469)

[3.5. Using indexes 6](#_Toc420657470)

[3.6. Backup strategy 6](#_Toc420657471)

[3.7. Monitoring and alarms tools 6](#_Toc420657472)

[3.8. To go further 6](#_Toc420657473)

[4. Conclusion 7](#_Toc420657474)

# Context

At Synchrotron Soleil, there are:

* One Tango database per beamline (29)
* One Tango database for the whole machine.

For database devices:

* On beamlines : 2 databases devices
* On the Machine : 20 database devices

A few numbers for the Machine:

select count(\*) from server; -> 4319

select count(\*) from device; -> 14846

select count(\*) from property\_device; -> 179075

select count(\*) from property\_attribute\_device; -> 184980

All these databases are still growing over the time. The service is stable and in operation since many years. But over the time, some problems have occurred. Since the Tango DB is a critical and central service, every small incident can quickly become a major incident and the whole control system become slow or unavailable.

# Problems

## Data unconsistency

Here some statistics on the Machine database:

* Servers without device:

select count(\*) from server where name not in (select \* from (select distinct server from device join server on device.server=server.name group by server) as s) ;

-> 1680

* Device without server

select count(\*) from server where name not in (select \* from (select distinct server from device join server on device.server=server.name group by server) as s) ;

-> 497

* Properties without devices:

select count(\*) from server where name not in (select \* from (select distinct server from device join server on device.server=server.name group by server) as s) ;

-> 30228

* Device property history without device:

select count(\*) from property\_device\_hist where device not in (select distinct name from device);

-> 155651

Deleting a device or a server does not delete everything. At Soleil, we have to manually « clean-up » the database. It is very difficult to maintain over the time. Moreover, every manual clean-up can lead to a risk of error and can break the whole database integrity.

## Performances and monitoring

A few months ago, an incident occurred and some slowness was reported by Soleil user during operation. After days of investigation, we found that the slowness was due:

* DbDeviceProperty and DbGetDeviceAttributeProperty commands are constantly used. These commands lock tables in databases.
* As a consequence, when a client tries to connect a device (DbImportDevice), some timeouts were thrown by the database device.

Example of timings (ms) found on some database devices:

|  |  |  |  |
| --- | --- | --- | --- |
| Command | average | minimum | maximum |
| DbGetDeviceAttributeProperty2 | 4051.671 | 0.050 | 120241.268 |
| DbPutDeviceAttributeProperty2 | 796.692 | 1.418 | 10745.142 |

Just for the record, the impact was considerable up to a point that the machine beam was becoming unstable. Some critical devices were stopping and could not be restarted without constantly restarting the database devices.

The root cause of the problem was finally a bug of the device database that was leading to fill a history table without cleaning it up.

But the main issue is that we do not have alarms on the database service, only a user notification (« Tango is slow ») rise an incident for the IT team. The incident should be detected well before the service availability is impacted. Moreover, we are missing diagnoses tools and procedures:

* + Impossible to activate logging to file on database devices
	+ no timing per SQL query
	+ no indication slow queries
	+ no indication on table contention
	+ …

Since Soleil has 31 tango databases on separated networks, it is mandatory for us to automate alarm reporting.

## Database backup

Today Soleil does one Mysql dump per day. During this dump, the database service in unavailable. Since the size of the database is growing, the dump time is also growing. Today, it takes up to 8 seconds on the Machine. At Soleil, the Machine fast orbit feedback has stopped many times because of this service interruption (the fast orbit feedback device has a memorized attribute).

# Soleil thoughts for reflection

## Advices

When developing a new device server, a developer must be aware on the impact on the database, for example:

* Do not use attribute memorization if not absolutely necessary.
* Do not write in device properties from device code on regular basis.

How many database devices should be deployed? Is it dependent on the number of devices?

This kind of advice should be clearly documented.

## Data consistency

The database schema may be reviewed to disable data inconsistency. It can be achieved by introducing constraints, cascades and triggers. Here is an example with the “Server” and “Device” tables with the use of primary keys, foreign keys and cascade delete:

CREATE TABLE `server` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

 PRIMARY KEY (id),

…

)

CREATE TABLE ` device` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

PRIMARY KEY (id),

` server\_id` int(11) NOT NULL,

FOREIGN KEY (server\_id) REFERENCES server(id) ON DELETE CASCADE,

 …

)

NB: not possible with MyISAM possible with InnoDb engine.

## Introducing ACID properties

To obtain ACID (<http://en.wikipedia.org/wiki/ACID>) properties, we could add transactions. Example:

BEGIN;

update table1;

update table2;

COMMIT; or ROLLBACK;

If something interrupts the transaction between two queries, the transaction is safe since all queries are roll backed.

NB: not all MySQL engines are transactional, MyISAM is not; InnoDB is.

Some others MySQL mechanisms could be introduction to avoid locking an entire table like [SELECT ... LOCK IN SHARE MODE](http://dev.mysql.com/doc/refman/5.0/en/select.html)

[SELECT ... FOR UPDATE](http://dev.mysql.com/doc/refman/5.0/en/select.html)

Cf. <http://dev.mysql.com/doc/refman/5.7/en/innodb-locking-reads.html>

## Audit and reporting

A module may be added to trace and report all changes in the database, so that we could, for example, track all deleted and added devices.

## Using indexes

To optimize performance on select queries, indexes can be created.

<https://dev.mysql.com/doc/refman/5.7/en/mysql-indexes.html>

## Backup strategy

 The backup operation should be totally transparent for the operation. Several options are possible:

* + MySQL replication: replicate the database and dump the replica that is not use for operation. The drawback is that the replication has to be closely monitored to obtain a good dump.
	+ MySQL provided paying backup tool (<https://www.mysql.fr/products/enterprise/backup.html>)
	+ Other strategy?

What is the best practice? This should be documented.

## Monitoring and alarms tools

To be able to easy detect defects, the database device should raise alarms when something goes wrong. It should also provide some key indicators for early diagnosis. This can be also archived by MySql monitoring tools, there are many of them:

* MySQL monitor (paying) <https://www.mysql.fr/products/enterprise/monitor.html>
* Some free tools: <http://www.tecmint.com/mysql-performance-monitoring>
* Nagios plugin: <http://www.nagios.com/solutions/mysql-monitoring>

What is the best practice? This should be documented.

## To go further

* SoC (Separation of Concerns): Tango DB provides several services such as device data persistency, data history and device directory. The services may be decoupled to simplify maintenance and operation.
* Compare the current solution with other databases like PostGreSQL (http://www.postgresql.org/) or NoSQL databases.

# Conclusion

SOLEIL operates Tango databases since more than 10 years. This document gives an overview of the problems raised by the current software architecture, design and implementation of the Tango database service.

A complete refactoring of this critical service would then be necessary.

SOLEIL does not have currently the development resources to take in charge this refactoring project. Nevertheless SOLEIL can actively participate in such project in the specifications and design phases and then be a test platform to validate on a real large scale installation the refactored solution.