Derby database tinkering with the Structure plugin database

TL;DR

If you ever have the misfortune of having to interrogate a Derby database, here are the takeaways:

- There's a primitive command-line utility called `ij`. Its interactive use can be made tolerable by running it repeatedly from the shell via: `ij -p < (echo "...connection properties...") <(echo "...sql commands...")`
- Once connected via `ij`:
  - `SHOW TABLES;` – shows tables
  - `DESCRIBE <tablename>;` – describes a particular table. If the tablename is uppercase, *don't* quote it here.
  - Quote uppercase table names in SQL queries. E.g. `SELECT "ID" from "SYNCS";`
- If ij's output is too narrow (row values end with '&' indicating truncation), run the SQL: `maximumdisplaywidth 10000;`
- For BLOBs, this page of examples is your friend.
- The Derby docs are quite good, but use frames, so Google will take you to some random-looking page without any context. Start from here.

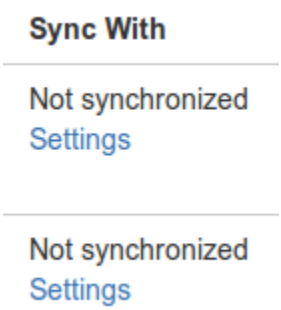
Introduction

The Structure plugin provides arbitrarily deep nesting of issues:

blocked URL

(nested issues are, incidentally, Atlassian's third-most popular feature request, JRA-4446. Go and vote!)

Twice now, after upgrading Structure, a client of mine found that the upgrade disables the auto-sync on Structures:



Nothing in the logs. While this gets worked out with ALMWorks support, I was asked to get things working again.

Command-line Derby spelunking

The Structure plugin's data is stored in its own embedded Derby database, in `$JIRA_HOME/structure/db/`. Derby is a fairly full-featured RDBMS written in Java.

First get the Derby distribution, which includes the `ij` command-line utility.

```
    wget 'http://archive.apache.org/dist/db/derby/db-derby-10.9.1.0/db-derby-10.9.1.0-bin.tar.gz'
tar zxvf db-derby-10.9.1.0-bin.tar.gz
export PATH+=:$HOME/db-derby-10.9.1.0-bin/bin
which ij
# /home/redradish/db-derby-10.9.1.0-bin/bin/ij
```

There is rather minimal documentation about ij and Derby generally.

Create an offline database copy

The first thing to know is, you can't connect to a Derby database if another process is using it. So no connecting to the live Structure database:

```
    ERROR XJ040: Failed to start database '/var/atlassian/application-data/jira/current/structure/db' with
class loader sun.misc.Launcher$AppClassLoader@409a44d6, see the next exception for details.
ERROR XSDB6: Another instance of Derby may have already booted the database /var/atlassian/application-data/jira
/6.4.9/structure/db.
```

So first we must make an offline copy of the Structure database;

```
    cp -ra /var/atlassian/application-data/jira/current/structure ~/structure_offline
```

Automate connecting to a database

`ij` is as primitive and ugly as you'd expect. There is exactly one way to invoke it: passing in a properties file for connection parameters, and a SQL file to run:

```
    ij --help
Usage: java org.apache.derby.tools.ij [-p propertyfile] [inputfile]
```

To auto-connect to a particular database, you need to create a properties file, and tell `ij` to use it. For instance I created a `~/structure_offline.props` file to connect to the `~/structure_offline/db` Derby database, copied above:

```
    echo "ij.database=jdbc:derby:${HOME}/structure_offline/db" > ~/structure_offline.props
```

and finally we connect:

```
    root@usw1-l-jira01:~# ij -p structure_offline.props
ij version 10.9
CONNECTION0* -     jdbc:derby:/home/redradish/structure_offline/db
* = current connection
ij>
```

Rather than create a pointless properties file on disk, the same thing can be achieved with a Bash process substitution:

```
    root@usw1-l-jira01:~# ij -p <(echo "ij.database=jdbc:derby:$HOME/structure_offline/db")
ij version 10.9
CONNECTION0* -     jdbc:derby:/home/redradish/structure_offline/db
* = current connection
ij>
```

Derby command-line Crash course

To list all tables:

```
    ij> SHOW TABLES;
TABLE_SCHEM         |TABLE_NAME                     |REMARKS
------------------------------------------------------------------------
SYS                 |SYSALIASES                     |
SYS                 |SYSCHECKS                      |
SYS                 |SYSCOLPERMS                    |
SYS                 |SYSCOLUMNS                     |
SYS                 |SYSCONGLOMERATES               |
SYS                 |SYSCONSTRAINTS                 |
SYS                 |SYSDEPENDS                     |
SYS                 |SYSFILES                       |
SYS                 |SYSFOREIGNKEYS                 |
SYS                 |SYSKEYS                        |
SYS                 |SYSPERMS                       |
SYS                 |SYSROLES                       |
SYS                 |SYSROUTINEPERMS                |
SYS                 |SYSSCHEMAS                     |
SYS                 |SYSSEQUENCES                   |
SYS                 |SYSSTATEMENTS                  |
SYS                 |SYSSTATISTICS                  |
SYS                 |SYSTABLEPERMS                  |
SYS                 |SYSTABLES                      |
SYS                 |SYSTRIGGERS                    |
SYS                 |SYSUSERS                       |
SYS                 |SYSVIEWS                       |
SYSIBM              |SYSDUMMY1                      |
APP                 |FAVORITES                      |
APP                 |HISTORY_V2                     |
APP                 |HISTORY_V2_ISSUES              |
APP                 |HISTORY_V2_PROJECTS            |
APP                 |LPROPS                         |
APP                 |PERSPECTIVES                   |
APP                 |PROPS                          |
APP                 |STATISTICS                     |
APP                 |STRUCTURES                     |
APP                 |STRUCTUREVIEWS                 |
APP                 |STRUCTUREVIEWS_V2              |
APP                 |SYNCS                          |
APP                 |VIEWS                          |
36 rows selected
```

Describe a particular table, in this case the SYNCS table:

```
    ij> DESCRIBE SYNCS;
COLUMN_NAME         |TYPE_NAME|DEC&|NUM&|COLUM&|COLUMN_DEF|CHAR_OCTE&|IS_NULL&
------------------------------------------------------------------------------
ID                  |INTEGER  |0   |10  |10    |NULL      |NULL      |NO
STRUCTUREID         |INTEGER  |0   |10  |10    |NULL      |NULL      |NO
MODULEKEY           |VARCHAR  |NULL|NULL|255   |NULL      |510       |NO
AUTOSYNCENABLED     |INTEGER  |0   |10  |10    |0         |NULL      |NO
USERNAME            |VARCHAR  |NULL|NULL|255   |NULL      |510       |YES
PARAMS              |BLOB     |NULL|NULL|21474&|NULL      |NULL      |YES
6 rows selected
```

Select a sample of values from a particular table. Note that the uppercase table names must be quoted, and there is no LIMIT clause unfortunately:

```
    ij> select "ID", "STRUCTUREID", "AUTOSYNCENABLED" from "SYNCS";
ID         |STRUCTUREID|AUTOSYNCEN&
-----------------------------------
100        |133        |0
101        |118        |0
102        |121        |0
....
205        |282        |1
206        |221        |1
98 rows selected
```

To deserialize a BLOB, in this case the PARAMS column:

(thanks to this doc)

At this point I got tired of the lack of readline support in `ij`, and switched to passing in SQL commands from the bash prompt:

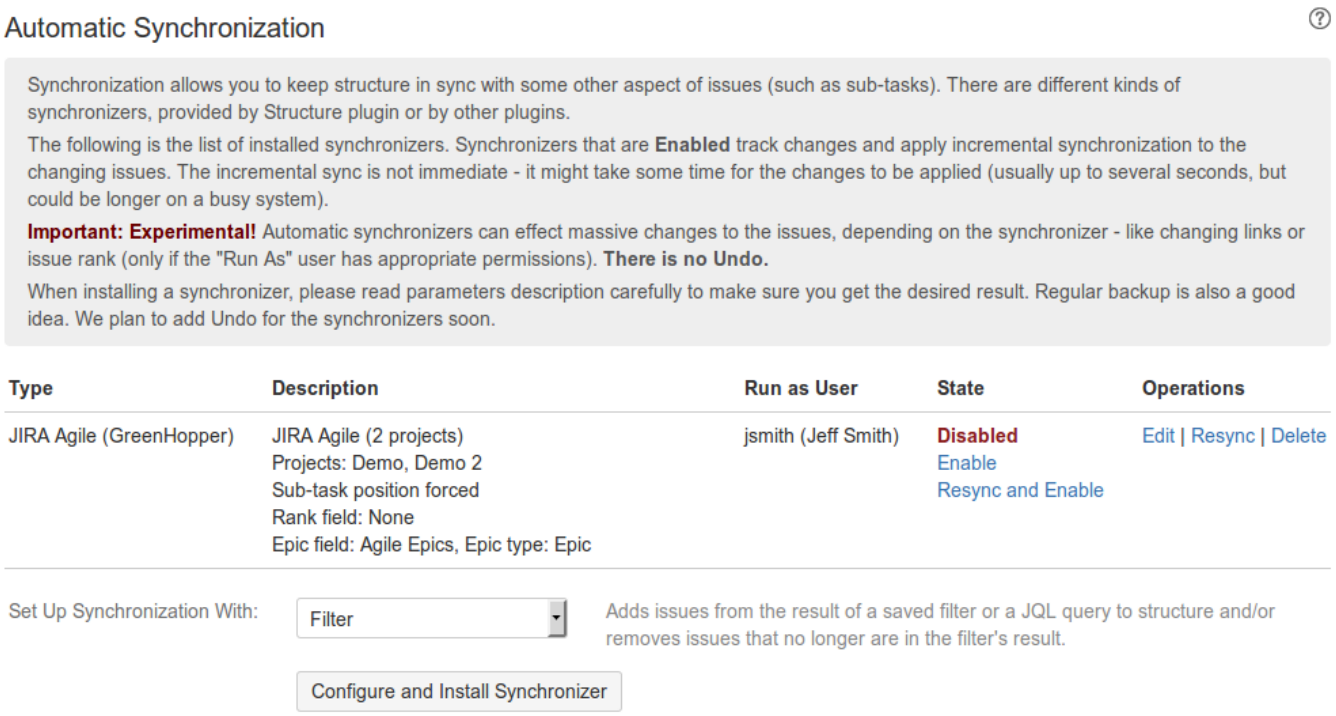
```
    ij> root@usw1-l-jira01:~# ij -p <(echo "ij.database=jdbc:derby:$HOME/structure_offlin") <(echo "select *
from SYNCS where STRUCTUREID=133")
ij version 10.9
CONNECTION0* -     jdbc:derby:/home/redradish/structure_offline/db
* = current connection
ij> select * from SYNCS where STRUCTUREID=133
;
ID
|STRUCTUREID|MODULEKEY
|AUTOSYNCEN&|USERNAME
|PARAMS

--------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------------------------------------------
--------------------------------------------------------------------------------
100        |133        |com.almworks.jira.structure:sync-
gh                                                                                                          |0
|jsmith
|3c3f786d6c2076657273696f6e3d22312e302220656e636f6de673d225554462d3822207374616e64616c6f6e653d22796573223f3e3c67
7265656e686f7&

1 row selected
```

Extracting BLOBs

In the result above, you can see the PARAMS BLOB is truncated (ending with '&' and isn't displaying anything human-readable. Some googling led me to th ese examples of the gloriously named SYSCS_UTIL.SYSCS_EXPORT_QUERY_LOBS_TO_EXTFILE function:

```
    root@usw1-l-jira01:~# rm -f /tmp/results.txt /tmp/resultlobs.txt; ij -p <(echo "ij.database=jdbc:derby:$HOME
/structure_offline/db") <(echo "maximumdisplaywidth 10000; CALL SYSCS_UTIL.SYSCS_EXPORT_QUERY_LOBS_TO_EXTFILE
('SELECT * FROM SYNCS WHERE STRUCTUREID=133', '/tmp/results.txt', '   ', '\"', 'UTF8', '/tmp/resultlobs.txt') ")
ij version 10.9
CONNECTION0* -     jdbc:derby:/home/redradish/structure_offline/db
* = current connection
ij> maximumdisplaywidth 10000;
ij> CALL SYSCS_UTIL.SYSCS_EXPORT_QUERY_LOBS_TO_EXTFILE('SELECT * FROM SYNCS WHERE STRUCTUREID=133', '/tmp
/results.txt', '    ', '"', 'UTF8', '/tmp/resultlobs.txt')
;
0 rows inserted/updated/deleted
root@usw1-l-jira01:~# cat /tmp/results.txt 100    133    "com.almworks.jira.structure:sync-gh"    0
"jsmith"    "/tmp/resultlobs.txt.0.291/"
root@usw1-l-jira01:~# cat /tmp/resultlobs.txt | xmllint -format -<?xml version="1.0" encoding="UTF-8"
standalone="yes"?>
<greenhopper-synchronizer>
  <projectId>0</projectId>
  <projects>
    <project>10044</project>
    <project>10050</project>
  </projects>
  <epicType>6</epicType>
  <forceSubtasks>true</forceSubtasks>
  <useEpicLinks>true</useEpicLinks>
</greenhopper-synchronizer>
root@usw1-l-jira01:~#
```

The parameter XML corresponds with what one sees at the Structure page in JIRA, https://jira.company.com/secure/StructureSync.jspa?id=133



This is all nice and interesting, but..

back to the task at hand.

Identifying disabled synchronizers

This displays a list of URLs for disabled synchronizers:

```
    root@usw1-l-jira01:~/structure# ij -p <(echo "ij.database=jdbc:derby:$HOME/structure_offline/db") <(echo
'select STRUCTUREID from SYNCS where AUTOSYNCENABLED=0 ') | grep '^[0-9]\+ *$' | while read id; do echo
"https://jira.example.com/secure/StructureSync.jspa?id=${id}"; done
https://jira.example.com/secure/StructureSync.jspa?id=133
https://jira.example.com/secure/StructureSync.jspa?id=118
https://jira.example.com/secure/StructureSync.jspa?id=121
https://jira.example.com/secure/StructureSync.jspa?id=129
https://jira.example.com/secure/StructureSync.jspa?id=109
....
```

Assuming we have

- the last-known-good Structure backup in `~/last_correctly_syncing_structure/`
- a recent copy of the actual Structure database in `/var/cache/rsnapshot/minutely.0/backup/var/atlassian/application-data/jira/current/structure`

this hacked-together script reports URLs of Structures we need to fix:

```
    #!/bin/bash -eu

ACTUAL_STRUCTURE_PATH=/var/cache/rsnapshot/minutely.0/backup/var/atlassian/application-data/jira/current
/structure
CORRECT_STRUCTURE_PATH=~/last_correctly_syncing_structure
export URL='https://jira.example.com/secure/StructureSync.jspa?id=%d
'

ijdb()
{
        ij -p <(echo "ij.database=jdbc:derby:db") $*
}
ij_allsync()
{
                ijdb <(echo 'select "STRUCTUREID", "AUTOSYNCENABLED" from "SYNCS" order by 1;')
}

export HOME=/home/redradish
export PATH=~/db-derby-10.9.1.0-bin/bin/:$PATH
mkdir -p ~/structure
rm -f ~/structure/{actual,correct}
cd $ACTUAL_STRUCTURE_PATH
ij_allsync | grep "^[0-9]" | sed -e 's/ *|/\t/g' > ~/structure/actual
cd $CORRECT_STRUCTURE_PATH
ij_allsync | grep "^[0-9]" | grep -v "rows selected" | sed -e 's/ *|/\t/g' > ~/structure/correct
echo "The following Structures used to auto-sync, and now do not auto-sync:"
join -a1 -a2 ~/structure/{correct,actual} | awk '$2==1 && $3==0 { printf ENVIRON["URL"], $1}'  | uniq
echo
echo "The following Structures auto-sync, and previously did not auto-sync:"
join -a1 -a2 ~/structure/{correct,actual} | awk '$2==0 && $3==1 { printf ENVIRON["URL"], $1}'  | uniq

root@usw1-l-jira01:~/structure# ./structure_syncs.sh
The following Structures used to auto-sync, and now do not auto-sync:
https://jira.example.com/secure/StructureSync.jspa?id=130
https://jira.example.com/secure/StructureSync.jspa?id=146
https://jira.example.com/secure/StructureSync.jspa?id=170
https://jira.example.com/secure/StructureSync.jspa?id=184
....
https://jira.example.com/secure/StructureSync.jspa?id=277

The following Structures auto-sync, and previously did not auto-sync:
https://jira.example.com/secure/StructureSync.jspa?id=170
root@usw1-l-jira01:~/structure
```

The identified Structures can then be fixed by hand.