Comunicado Técnico 118
ISSN 1677-8464
Dezembro, 2015
Campinas, SP

Versionamento de banco de dados com a ferramenta Liquibase: aplicação ao módulo web do Sistema de Informação de Experimentos da Embrapa (SIExp)

Daniel Rodrigo de Freitas Apolinário[1]
Leonardo Ribeiro Queiros[2]
Sergio Aparecido Braga da Cruz[3]

O controle de versões combina procedimentos e ferramentas para gerenciar diferentes versões dos objetos de configuração criados durante o processo de software (PRESSMAN, 2011). O versionamento de artefatos de software é uma prática já bastante consolidada principalmente quando se tem em mente o código fonte de um sistema. Por outro lado, os objetos de configuração relacionados a banco de dados de um sistema possuem características distintas de um código fonte textual tradicional, uma vez que podem especificar modificações estruturais na organização dos dados e que podem implicar em um esforço de migração de dados já armazenados no banco. O uso de técnicas de versionamento tradicionais neste caso pode causar problemas relacionados à falta de sincronização entre a versão da aplicação e a versão do banco de dados, assim como um trabalho muito grande e manual para garantir a consistência da versão correta da base de dados com a versão do software.

A abordagem de utilizar ferramentas tradicionais de controle de versão de software para os artefatos relacionados ao banco de dados se mostra ineficiente, uma vez que a evolução incremental do banco de dados é estruturalmente diferente da evolução de código fonte e ao mesmo tempo é um requisito fundamental para a gestão e manutenção de um sistema no ambiente de produção.

Uma das maneiras de tratar esta questão é elaborar um processo manual de versionamento do banco de dados e incorporá-lo ao processo de desenvolvimento. Outra forma seria usar alguma ferramenta para automatizar parte deste controle de versão de banco de dados. Como o processo manual acaba dependendo muito do fator humano, e que por isso pode ser mais propenso a erros, a abordagem de utilizar uma ferramenta específica vem sendo bastante utilizada em projetos de desenvolvimento de software.

Dentre as ferramentas *open-source* existentes no mercado que automatizam o controle de versionamento de banco de dados, foi feita uma pesquisa na documentação das ferramentas Flyway (FLYWAY, 2010) e Liquibase (LIQUIBASE, 2007). Não é o objetivo deste trabalho a comparação entre as duas ferramentas citadas, já que não foi realizado um estudo comparativo detalhado entre as elas e sim foi escolhida a ferramenta Liquibase por apresentar maior simplicidade na instalação e execução, e pelo fato de ser suportada pela

[1] Cientista da computação, especialista em Plataformas de Desenvolvimento Web, analista da Embrapa Informática Agropecuária, Campinas, SP.
[2] Cientista da computação, doutor em Engenharia Agrícola, analista da Embrapa Informática Agropecuária, Campinas, SP.
[3] Engenheiro eletricista, doutor em Computação Aplicada, pesquisador da Embrapa Informática Agropecuária, Campinas, SP.

ferramenta SQL Power Architect que é bastante utilizada para a modelagem de banco de dados.

Este trabalho apresenta a ferramenta Liquibase para automação do processo de gestão de configuração de objetos relacionados à construção de bancos de dados.

Para facilitar a compreensão do leitor, este trabalho foi divido em três seções. Na primeira seção, a ferramenta Liquibase é apresentada na forma de suas funcionalidades. Em seguida, é relatada a aplicação do Liquibase no projeto do SIExp descrevendo a integração da ferramenta ao projeto, o novo processo de geração de mudanças de banco de dados, as limitações e dificuldades encontradas na sua adoção. Por último, são apresentados os principais resultados e contribuições obtidos com a execução deste trabalho.

1 Apresentação do Liquibase

1.1 Funcionalidades

As principais funcionalidades da ferramenta Liquibase estão listadas a seguir:

- *Update*: Aplicação de mudanças, quer seja de estrutura ou de dados.
- *Rollback*: Desfaz mudanças de banco de maneira automática ou programada.
- *Diff*: Verifica diferenças entre duas versões de banco de dados.
- *SQL Output*: Não aplica diretamente as mudanças em um banco de dados, mas gera scripts. SQL com todas as mudanças a serem aplicadas.

O Liquibase possui integração com vários tipos de Sistemas Gerenciadores de Banco de Dados (SGBD's), dentre os quais Oracle, PostgreSQL, MySQL, SQL Server, SQLite. Também possui integração com algumas ferramentas e frameworks como Ant, Maven, Spring etc.

Os arquivos de mudanças do Liquibase podem ser escritos em diversas linguagens tais como: XML, YAML, JSON e SQL. De acordo com o SGBD utilizado, o Liquibase "traduz" a linguagem na qual as mudanças foram escritas para a linguagem original do SGBD antes de aplicar as mudanças no banco de dados do usuário.

1.2 Conceitos

A seguir são apresentados os principais conceitos do Liquibase:

- *Arquivo de Changelog*: Este é o arquivo principal do Liquibase, pois é lido pelo executor do Liquibase toda vez que a ferramenta é executada. Neste arquivo são listadas todas as pré-condições, atributos e conjunto de mudanças que serão realizadas no banco de dados.

- *ChangeSet*: Como o próprio nome sugere, é o conjunto de mudanças a serem aplicadas em um banco de dados. Estas mudanças podem ser na estrutura do banco e/ou em seus dados. Um arquivo de *changelog* deve conter um ou mais *changesets*. Normalmente, este conjunto de mudanças deve ser agrupado de tal maneira que elas devem ser atômicas, isto é, ou todas as mudança são aplicadas com sucesso ou no caso de alguma falhar, nenhuma delas deve ser aplicada. Cada *changeset* deve ser identificado unicamente por um "*id*" e um "*author*", pois esta é a sua chave para o Liquibase.

- *Preconditions*: São verificações realizadas com o intuito de decidir sobre a aplicação ou não de mudanças. Pode haver pré-condições tanto no arquivo de *changelog* e que se aplicam a todos os changesets que fazem parte dele, quanto especificamente para um determinado *changeset*.

- *Contexts*: São marcações que podem ser usadas para definir em quais ambientes os *changesets* serão ou não executados. O exemplo mais simples de uso desta funcionalidade é para diferenciar a execução para os diferentes ambientes de desenvolvimento de um software, tais como produção, teste e desenvolvimento.

1.3 Funcionamento

Para controlar a execução dos *changesets*, o Liquibase utiliza duas tabelas relacionais (Databasechangelog, Databasechangeloglock) que podem ser criadas manualmente ou então são criadas automaticamente na primeira vez que a ferramenta é executada. Na configuração da ferramenta pode-se indicar qual o banco e *schema* que estas tabelas serão criadas, ficando a cargo do usuário decidir se este controle pode ficar no mesmo banco de dados que se está versionando ou em outro banco de dados separado. Na Tabela 1,

Tabela 1. Estrutura e a descrição traduzida dos metadados da tabela relacional Databasechangelog.

Coluna	Tipo do dado	Descrição
ID	VARCHAR(255)	Valor do atributo "*id*" do *changeset*
AUTHOR	VARCHAR(255)	Valor do atributo "*author*" do *changeset*.
FILENAME	VARCHAR(255)	*Path* do arquivo de *changelog*. Pode ser um *path* absoluto ou relativo.
DATEEXECUTED	DATETIME	Data/hora da execução do *changeset*. É usado com ORDEREXECUTED para determinar a ordem de *rollback*.
ORDEREXECUTED	INT	Numérico que indica a ordem que o *changeset* foi executado. Usado em conjunto com DATEEXECUTED para garantir a ordenação da execução dos *changesets*. Observação: Somente há a garantia de que os valores são incrementais dentro de uma mesma execução de um update.
EXECTYPE	VARCHAR(10)	Descrição de como o *changeset* foi executado. As opções possíveis são: "EXECUTED", "FAILED", "SKIPPED", "RERAN", e "MARK_RAN".
MD5SUM	VARCHAR(35)	*Checksum* do *changeset* no momento que ele foi executado. Usado em toda execução para garantir que não há *changesets* que foram alterados depois de já terem sido executados.
DESCRIPTION	VARCHAR(255)	Pequena descrição do *changeset* que é autogerada.
COMMENTS	VARCHAR(255)	Valor do atributo "*comment*" do *changeset*.
TAG	VARCHAR(255)	Usada para rastrear quais *changesets* correspondem à quais *tags*.
LIQUIBASE	VARCHAR(20)	Versão do Liquibase que foi utilizada na execução do *changeset*.

Fonte: Liquibase (2015), adaptada pelo autor.

é apresentada a estrutura e a descrição traduzida dos metadados da tabela relacional Databasechangelog.

A Tabela Databasechangeloglock não será demonstrada aqui, mas ela é utilizada para controlar o lock da tabela que armazena as mudanças de maneira a não permitir que ocorram execuções concorrentes do Liquibase em um mesmo banco de dados.

Resumidamente, o funcionamento do Liquibase acontece da seguinte maneira. O arquivo *changelog* é lido e para cada *changeset* contido nele, é feita uma verificação se este conjunto de mudanças já foi executado. Caso já tenha sido executado, o Liquibase pula para o próximo *changeset*, caso contrário, o Liquibase tentará executar as mudanças contidas nele. Quando a ferramenta consegue aplicar com sucesso o conjunto de mudanças, a Tabela Databasechangelog é atualizada com o registro da execução daquele *changeset* de maneira que na próxima execução do Liquibase a mudança não mais seja aplicada, fazendo assim um versionamento incremental de todas as mudanças relacionadas ao banco de dados.

Desta maneira, o Liquibase pode responder quem, quando e onde foram aplicadas cada mudança no banco de dados, além de possibilitar desfazer mudanças até um determinado *changeset* (neste caso é necessário que cada conjunto de mudanças tenha uma seção que implemente o seu "*rollback*"). Também é possível saber facilmente quais as diferenças entre bancos de dados de diferentes ambientes, já que basta identificar quais são os *changesets* que estão aplicados em cada um dos ambientes.

2 Aplicação do Liquibase no projeto do SIExp

A adoção da ferramenta Liquibase foi realizada durante o desenvolvimento do módulo web do Sistema de Informação de Experimentos (SIExp), da Empresa Brasileira de Pequisa Agropecuária (Embrapa), desenvolvido no âmbito do projeto MP5 'Gestão dos dados experimentais da Embrapa' – projeto SIExp. Este sistema foi construído utilizando tecnologias *open-source* da plataforma de desenvolvimento Java. O banco de dados utilizado foi o SGBD PostgreSQL versão 9.3. Para a modelagem do banco de dados, foi utilizada a ferramenta SQL Power Architect Community Edition.

O processo de desenvolvimento do SIExp é baseado em técnicas ágeis adapdatas do Scrum (SCRUM, 2009) no qual as funcionalidades do sistema foram entregues de forma iterativa e incremental. Neste processo, o modelo do banco de dados era alterado a cada nova iteração, porém, a execução desta evolução seguia a lógica de recriar totalmente a base de dados. No início da fase de desenvolvimento do SIExp, não havia controle de versão do banco de dados, ou seja, a cada nova alteração efetuada no modelo de dados, o desenvolvedor deveria executar uma tarefa da ferramenta Ant (ANT, 1999) que primeiramente recriava toda a estrutura do banco de dados e em seguida inseria novamente todos os dados de carga da aplicação. Este processo apresentava dois problemas, sendo que o principal era o tempo de demora na execução completa dos scripts de carga. O outro problema detectado era a perda de dados de teste que ocorria no processo que apagava toda a estrutura e dados de sua base de da-

dos local para a recriação do banco. Além disso, uma vez que o sistema estivesse no modo produção, era fundamental que houvesse uma solução para a realização de alterações incrementais no banco de dados, já que neste caso os dados precisam ser sempre preservados. Estes problemas levaram à equipe do projeto a incorporar ao projeto uma ferramenta para automatizar o controle de versão de seu banco de dados.

2.1 Integrando o Liquibase ao projeto

Optou-se, inicialmente, por utilizar a integração fornecida pela ferramenta SQL Power Architect (Figura 1), de maneira a facilitar a geração dos *changesets*, pois, no processo de desenvolvimento do projeto, qualquer mudança de banco de dados é feita primeiramente no modelo usando a referida ferramenta. Porém, num primeiro teste, avaliou-se que a integração permitia gerar somente XML e mesmo assim este código gerado não era executado corretamente pelo Liquibase. Portanto, decidiu-se abortar o uso da integração com o SQL Power Architect e a preferência foi em usar a opção de gerar *changesets* em SQL, para não correr o risco de haver situações nas quais o projeto teria uma parte de changeset em XML e parte em SQL, o que não era desejado.

Também foi feita a opção de utilizar somente um arquivo de *changelog* e, para cada *changeset* criar um arquivo separado somente com o código SQL correspondente às mudanças. Procurou-se, desta maneira, isolar os códigos SQL do código XML de controle do *changelog*. Na Figura 2 pode-se observar um arquivo de changelog similar ao que foi criado para o SIExp. Na Figura 3 é exibido um exemplo fictício de um arquivo de *changeset* que tem apenas uma mudança.

Como já fora dito neste documento, antes de se adotar o Liquibase, cada mudança no banco de dados gerava uma alteração num script que excluía o banco de dados e o recriava inteiramente. Além disso, o projeto também

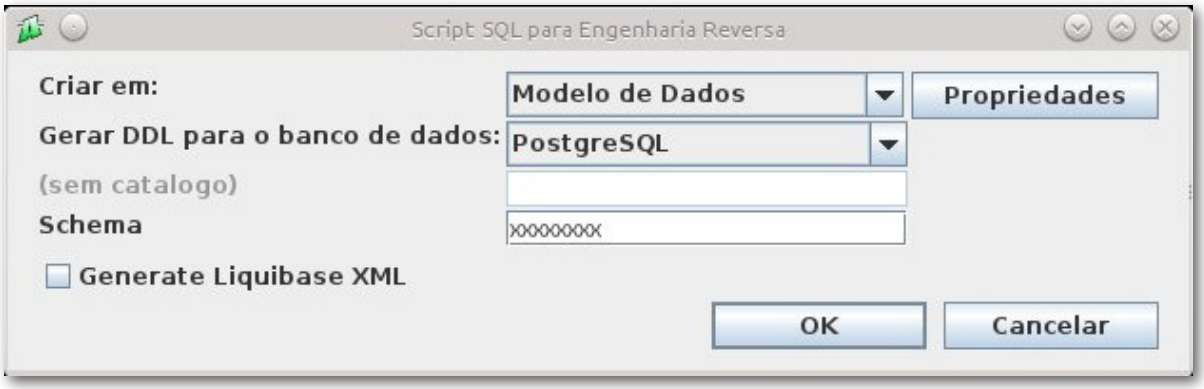

Figura 1. Tela do SQL Power Architect onde há a opção de gerar arquivo XML do Liquibase.

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog
        xmlns="http://www.liquibase.org/xml/ns/dbchangelog"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ext="http://www.liquibase.org/xml/ns/dbchangelog-ext"
        xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-3.1.xsd
        http://www.liquibase.org/xml/ns/dbchangelog-ext http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-ext.xsd">

        <changeSet author="desenvolvedor_1" id="1" logicalFilePath="changeset_1">
                <preConditions onFail="MARK_RAN">
                        <not>
                                <changeLogPropertyDefined property="environment" value="Production"/>
                        </not>
                </preConditions>
                <comment>Scripts de criação do banco</comment>
                <sqlFile path="sql/script_inicial_1.sql" relativeToChangelogFile="true"/>
                <sqlFile path="sql/script_inicial_2.sql" relativeToChangelogFile="true"/>
        </changeSet>

        <changeSet author="desenvolvedor_1" id="2" logicalFilePath="changeset_2.sql">
                <comment>Inserção de dados de tabelas de domínio</comment>
                <sqlFile path="changeset_2.sql" relativeToChangelogFile="true"/>
        </changeSet>

        <changeSet author="desenvolvedor_2" id="3" logicalFilePath="changeset_3.sql">
                <comment>Nova alteração no modelo</comment>
                <sqlFile path="changeset_3.sql" relativeToChangelogFile="true"/>
        </changeSet>

</databaseChangeLog>
```

Figura 2. Arquivo de *changelog*.

```
-- Cria uma nova coluna na tabela cliente

ALTER TABLE schema_clientes.cliente ADD COLUMN data_validade_contrato TIMESTAMP;
```

Figura 3. Exemplo de um arquivo de *changeset.*

possuía scripts separados que faziam carga de dados iniciais para a execução da aplicação. Com a implantação do Liquibase, este método anterior tinha de ser tratado de alguma maneira. E, a solução foi criar um *changeset* para os scripts Data Definition Language (DDL) que eram responsáveis por criar toda a estrutura do banco de dados, até então. Este *changeset* foi configurado como “MARK_RAN”, o que significa que o Liquibase deve considerar que o *changeset* foi executado, porém não executá-lo de fato. Também foi criado outro *changeset* configurado como “MARK_RAN”, que contém todos os scripts Data Manipulation Language (DML) que eram responsáveis pela carga inicial dos dados no sistema. Desta maneira, caso alguém queira recriar totalmente o banco de dados do SIExp usando o Liquibase pode fazê-lo desde que se retire a tag “MARK_RAN” destes dois *changesets* citados anteriormente.

O Liquibase pode ser executado de várias maneiras e, dentre elas, via linha de comando ou pela ferramenta Ant. Como o projeto do SIExp utiliza o Ant para a automatização de build, optou-se por criar uma task no Ant, chamada “atualizadb”, que é responsável por executar o Liquibase no banco de dados do projeto. A Figura 4 mostra o código do Ant similar ao que foi implementado para integrar o Liquibase ao nosso processo de automatização de build.

A utilização do Ant para a execução do Liquibase resolveu o problema relacionado às máquinas dos desenvolvedores, porém, para as máquinas de homologação e produção, a solução escolhida foi a de executar o Liquibase por meio de sua interface via linha de comando. Para isso, foi criado um arquivo de propriedades nomeado “liquibase.properties” o qual contém todas as propriedades que o Liquibase precisa para acessar o banco de dados no qual ele aplicará as mudanças, tais como driver de conexão, nome do banco de dados, porta de acesso, usuário e senha do banco de dados. Como o projeto já possuía um script *Shell* do Linux para a instalação de um pacote da aplicação, ele foi modificado para invocar a biblioteca do Liquibase via linha de comando, passando os parâmetros necessários para a sua execução.

2.2 Processo de geração de mudanças de banco de dados

Após a implantação do Liquibase no projeto do SIExp, a principal consequência para os desenvolvedores foi

```
<taskdef resource="liquibase/integration/ant/antlib.xml" uri="antlib:liquibase.integration.ant">
        <classpath path="${liquibase.home}/liquibase.jar"/>
</taskdef>

<liquibase:database
        id="MEUBANCODB"
        driver="org.postgresql.Driver"
        url="jdbc:postgresql://localhost:5432/{nome do meu banco de dados}"
        user="{nome do meu usuário do banco de dados}"
        password="{senha do meu usuário de banco de dados}"
        defaultSchemaName="{schema do meu banco de dados}"
/>


<target name="atualizadb" >
        <liquibase:updateDatabase loglevel="debug"
                changelogfile="/meuchangelog.xml"
                databaseref="MEUBANCODB" >
                <classpath refid="postgresql.driver.classpath"/>
                <liquibase:changeLogParameters>
                                <liquibase:changeLogParameter name="environment" value="Production"/>
                                <propertyset>
                                        <propertyref prefix="params"/>
                                </propertyset>
                        </liquibase:changeLogParameters>
        </liquibase:updateDatabase >
</target>
```

Figura 4. Código do script de build da ferramenta Ant integrada com o Liquibase.

a alteração no processo de geração de mudanças de banco de dados. Os passos a serem seguidos quando se deseja alterar o modelo de banco de dados são os seguintes:

1) Efetuar a alteração necessária no modelo de dados usando o SQL Power Architect.

2) Comparar o modelo de dados alterado com o original (antes da mudança) por meio da funcionalidade de comparação de modelos do próprio SQL Power Architect. Esta funcão gera um script SQL para "transformar" o modelo original no novo modelo alterado.

3) Criar um arquivo de *changeset* que irá conter um script com o código SQL gerado no passo anterior.

4) Caso necessário, fazer alterações no script SQL gerado no passo anterior.

5) Criar um nome único para este arquivo de *changeset* gerado no passo 4 (sugestão: usar o id do *changeset* e um *timestamp*).

6) Incluir o *changeset* no arquivo de *changelog*, configurando adequadamente as propriedades de "author" e "id".

7) Executar o Liquibase pela *task* "atualizadb" do Ant.

8) Verificar se as mudanças foram aplicadas corretamente na base de dados.

9) Versionar o modelo e os arquivos de *changelog* e *changeset* no controlador de versões SVN.

O passo 4 pode ser necessário porque o script gerado pelo SQL Power Architect só considera o modelo de dados, e na verdade, a mudança terá que considerar também os dados já existentes no banco. Um exemplo claro desta situação acontece quando é feita uma alteração de uma coluna de NULLABLE para NOT NULL. Neste caso, o script gerado pelo SQL Power Architect só gera o código que faz a alteração no tipo da coluna, porém ao executar este script em um banco de dados que já contenha linhas nesta tabela e possua valores nulos nesta coluna, ele não irá funcionar devido às restrições de integridade do banco de dados. Deste modo, as linhas que têm valores nulos precisam ser apagadas ou preenchidas com seus respectivos valores antes de se mudar o tipo dela para NOT NULL.

Se a alteração no banco de dados contiver mudanças somente nos dados (por exemplo, um script de carga de dados ou de atualização de dados), o processo inicia no passo 3, sendo que o script SQL será gerado manualmente pelo desenvolvedor.

Foi definido que o atributo "id" do *changeset*, seria um sequencial para facilitar a identificação da ordem cronológica das alterações de banco de dados, porém, como o trabalho é concorrente e em equipe, não há garantias de que este id seja único. Neste caso, a chave única é a composição do atributo "author" e "id".

Após a execução do Liquibase, no passo 7, ao consultar a tabela Databasechangelog pode-se verificar o registro que a ferramenta insere referente ao changeset que acaba de ser executado (Figura 5).

2.3 Limitações e dificuldades

O processo de instalação e configuração do Liquibase ocorreu sem maiores problemas. Uma dificuldade encontrada neste processo foi configurar o Ant para executar o Liquibase, mas foi superada após investigação de problemas semelhantes em fóruns de usuários na internet.

Uma desvantagem percebida, após a implantação do Liquibase no SIExp, foi a perda da garantia de que o modelo de dados gerado e atualizado na ferramenta SQL Power Architect reflete exatamente a base de dados dos ambientes dos projetos. Isto ocorre devido ao fato de que um erro do desenvolvedor (exemplo: não atualizar adequadamente o modelo ou gerar um script SQL que não reflita exatamente a mudança no modelo) pode causar inconsistência entre o modelo de dados e a base de dados real, pois a base de dados agora só depende dos *changesets* gerados. Esta probabilidade de inconsistência não ocorria antes, já que o script de geração da estrutura do banco de dados era sempre construído com base no modelo do SQL Power Architect.

No início da adoção do Liquibase, foi definido que os nomes dos arquivos de changeset teriam um padrão que incluiria o id do *changeset* para torná-los únicos,

	id character varying(255)	author character varying(255)	filename character varying(255)	dateexecuted timestamp without time zone	orderexecuted integer	exectype character varying(10)	md5sum character varying(35)	description character varying(255)	comments character varying(255)	tag character varying(255)	liquibase character varying(20)
1	1100	desenvolvedor_1	changeset_1100	2015-11-18 00:00:00	100	EXECUTED	7:c5d00002fc51df	sqlFile	Criação de nova c		3.3.0

Figura 5. Resultado de consulta à tabela Databasechangelog.

porém, no decorrer do projeto foi verificado que poderiam ocorrer nomes iguais dos arquivos de *changeset*, no caso de desenvolvedores diferentes concorrentemente criarem um conjunto de mudanças com o mesmo id. A partir deste momento, definiu-se que o nome do arquivo deveria conter também um "*timestamp*" indicando data/hora que o arquivo foi gerado.

Uma limitação verificada na ferramenta tem a ver com a falta de informações do log de erros de execução do Liquibase. Muitas vezes é difícil identificar a causa de algum erro de execução porque o log não contém muita informação que aponta diretamente o erro. No entanto, os erros de execução mais comuns observados foram erros no próprio código SQL (que pode ser reproduzido se for executado diretamente numa ferramenta cliente de banco de dados como o Pgadmin) ou erros referentes à verificação de *checksum* do arquivo, caso ele tenha sido alterado depois de já ter sido executado.

3 Conclusão

A adoção da ferramenta Liquibase no projeto do SIExp trouxe algumas dificuldades iniciais, principalmente no que tange à adaptação da equipe no novo processo de trabalho, porém, trouxe muitos benefícios no que se refere à atualização incremental da base de dados. O processo de atualização da base de dados ficou muito rápido para o desenvolvedor, pois, enquanto no processo anterior teria de esperar vários minutos para que o banco de dados fosse recriado e repopulado, no novo processo incremental a atualização demora poucos segundos. Os dados utilizados nas bases de dados locais de desenvolvimento também não precisam mais ser perdidos com o novo processo. E ainda continua disponível a opção de que o desenvolvedor recrie totalmente a base de dados caso seja necessário e/ou desejado.

A experiência mostrou que o Liquibase trouxe mais agilidade e flexibilidade para o ambiente de desenvolvimento do projeto do SIExp e também atendeu à necessidade de atualizações incrementais de banco de dados no ambiente de produção e, desta forma, recomenda-se o uso deste tipo de ferramenta em projetos que tenham as mesmas necessidades apontadas pelo projeto do SIExp.

4 Referências

ANT. 1999. Disponível em: <http://ant.apache.org/>. Acesso em: 9 out 2015.

FLYWAY. Disponível em: <http://flywaydb.org/>. Acesso em: 9 out 2015.

LIQUIBASE. 2007. Disponível em: <http://www.liquibase.org/documentation/index.html>. Acesso em: 9 out. 2015.

LIQUIBASE. **Databasechangelog table**. 2015. Disponível em: <http://www.liquibase.org/documentation/databasechangelog_table.html>. Acesso em: 9 out. 2015.

PRESSMAN, R. S. **Engenharia de software**: uma abordagem profissional. 7. ed. Porto Alegre: Bookman, 2011, p. 475.

SCRUM. 2009. Disponível em: <https://www.scrum.org/>. Acesso em: 9 out. 2015.

Comunicado Técnico, 118

Embrapa Informática Agropecuária
Endereço: Caixa Postal 6041 - Barão Geraldo
13083-886 - Campinas, SP
Fone: (19) 3211-5700
www.embrapa.br/informatica-agropecuaria
sac: www.embrapa.br/fale-conosco/sac/

Embrapa
Ministério da Agricultura, Pecuária e Abastecimento
GOVERNO FEDERAL BRASIL PÁTRIA EDUCADORA

1ª edição publicação digital - 2015

Expediente

Supervisão editorial: *Stanley Robson de Medeiros Oliveira, Neide Makiko Furukawa*

Normalização bibliográfica: *Maria Goretti Gurgel Praxedes*

Revisão de texto: *Adriana Farah Gonzalez*

Editoração eletrônica/Arte final: *Neide Makiko Furukawa*

CGPE 12470