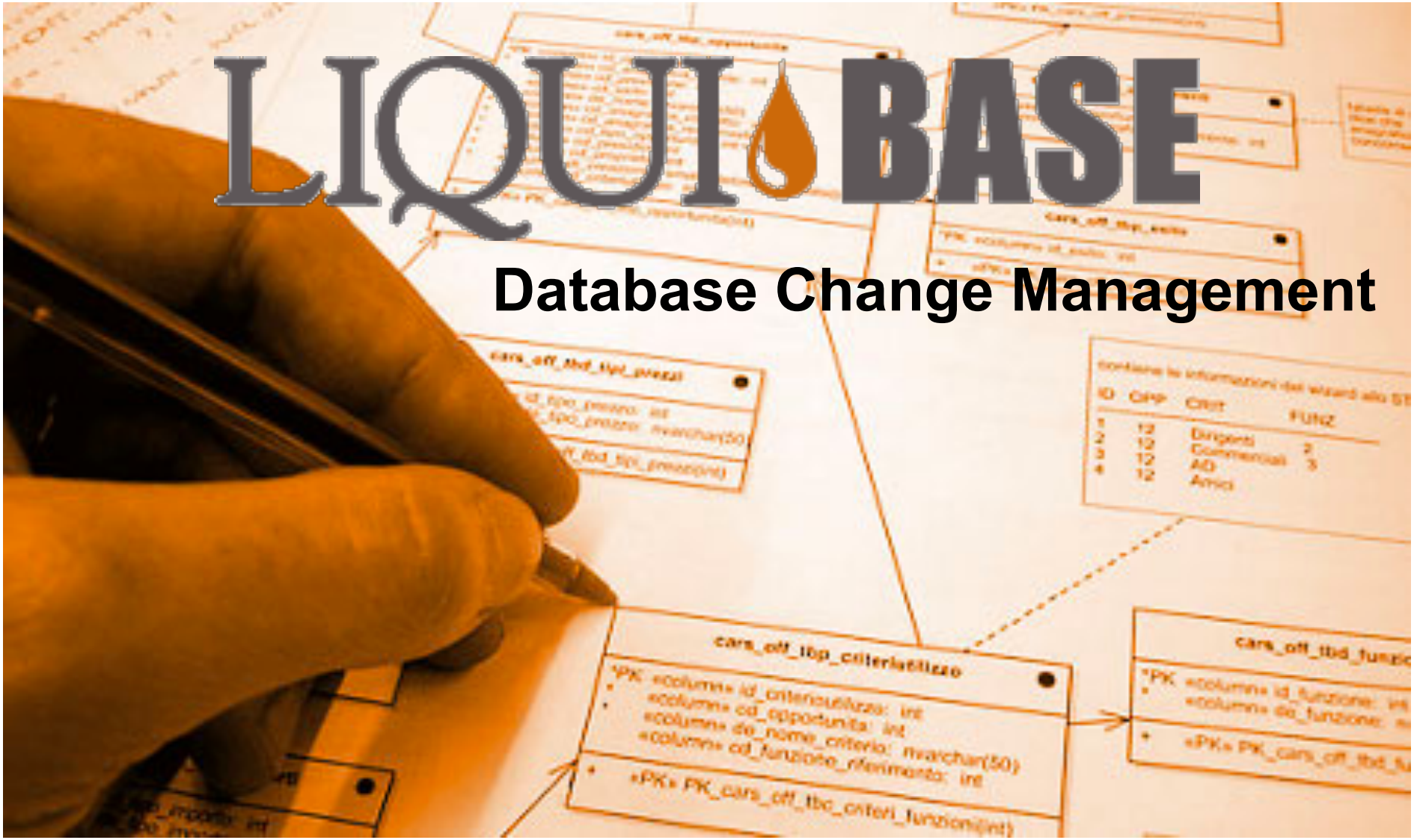


Rikard Thulin, Squeed

About the Presenter

Rikard Thulin

No association to Liquibase except as a pleased user

User group leader for Javaforum Gothenburg and project lead for the community arranged conference dev:mobile 2012, 2013 and 2014.

Co-founder and owner of Squeed.

The full presentation: http://goo.gl/ENjiHY

e-mail: rikard.thulin@squeed.com, twitter: @rikard_thulin, blog: http://squeed.com/blog

The Question?

You never develop code without version control,

why would you develop your database without it?

Liquibase at a glance

Liquibase is an Apache 2.0 Licensed tool for tracking, managing and applying database changes

Database changes are stored in an XML (or JSON, YAML, plain-ish SQL) file that is version controlled and database independent

Liquibase managed and applies changes to the database

Major Concepts - Changeset

```
<changeSet author="liquibase-docs" id="createTable-example">
    <createTable catalogName="cat"
            remarks="A String"
            schemaName="public"
            tableName="person"
            tablespace="A String">
        <column name="address" type="varchar(255)"/>
    </createTable>
</changeSet>
```

Major Concepts - Changeset

A changeset is uniquely identified change that should be applied to the database

When Liquibase runs, it queries the DATABASECHANGELOG table for the changesets that are marked as executed and then executes all changesets that have not yet been executed

Liquibase feature set

- Automatic rollbacks
- Database ”diff“s
- Generating starting change logs from existing databases
- Generating database change documentation
- Code branches and merging

Supported Refactorings

Structural Refactorings

- Add Column
- Rename Column
- Modify Column
- Drop Column
- Alter Sequence
- Create Table
- Rename Table
- Drop Table
- Create View
- Rename View
- Drop View
- Merge Columns
- Create Stored Procedure

Data Quality Refactorings

- Add Lookup Table
- Add Not-Null Constraint
- Remove Not-Null Constraint
- Add Unique Constraint
- Drop Unique Constraint
- Create Sequence
- Drop Sequence
- Add Auto-Increment
- Add Default Value
- Drop Default Value

Referential Integrity Refactorings

- Add Foreign Key Constraint
- Drop Foreign Key Constraint
- Drop All Foreign Key Constraints
- Add Primary Key Constraint
- Drop Primary Key Constraint

Non-Refactoring Transformations

- Insert Data
- Load Data
- Load Update Data
- Update Data
- Delete Data
- Tag Database
- Stop

Architectural Refactorings

- Create Index
- Drop Index

Custom Refactorings

- Modifying Generated SQL
- Custom SQL
- Custom SQL File
- Custom Refactoring Class
- Execute Shell Command

Database Change Management

- Database deltas are part of the code change
- The correlation between code revisions are ensured
- Allows the automatisation of database changes
- Keeps all environments consistent
- Allows reverting a previous version of a system

Why Liquibase?

Production Node 1  Production Node 2

Staging Server Node 1  Staging Server Node 2

Test / QA server 

Development server  CI Server

8 developers with 8 local databases/schemas

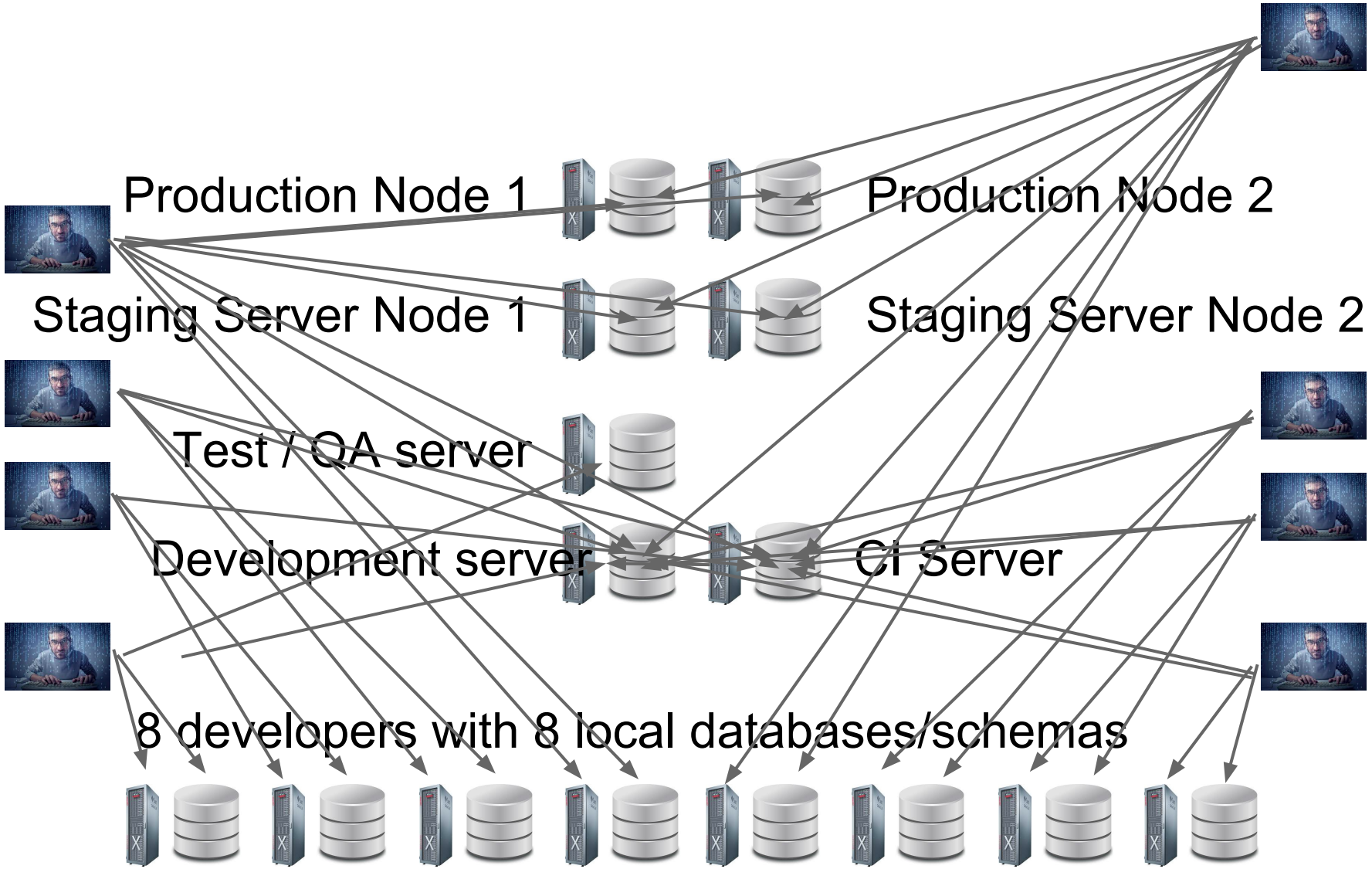
Old School

1. I need to add column A to table B
2. Develop code change and SQL
3. Add SQL to jira release issue
4. Commit code
5. Send e-mail to all developers to manually apply SQL to their local DB
6. Manually apply SQL to DEV and CI environment
7. Manually deploy code to DEV and CI environment

Old School

8. Days later, read jira issue and manually apply SQL to QA environment
9. Deploy code to QA environment
10. Days later, read jira issue and manually apply SQL to STAGING environment
11. Deploy code to STAGING environment
12. Days later, read jira issue and manually apply SQL to PRODUCTION environment
13. Deploy code to PRODUCTION environment

Old School


Production Node 1
Production Node 2
Staging Server Node 1
Staging Server Node 2
Test / QA server
Development server
CI Server
8 developers with 8 local databases/schemas

Sure but we have automated….

Probably you have used the anti-pattern

“not invented here”

...to solve small parts, half elegant, as of what Liquibase (and other similar tools) does

The Liquibase ways

1. I need to add column A to table B
2. Develop code change and SQL
3. Commit code

Database changes are applied automatically by the container (spring, servlet listener or CDI) and is bundled with the deployment artifact.

Commit and forgett!

Demo

1. Update Database with pending changesets
2. Rollback previous changes
3. Generate rollback SQL

Tips and tricks - wrong is right

One of the most difficult challenges is to overcome “wrong is right” principle

- When a changeset has been committed, it can not be modified even if it is “wrong”
- You must write another changeset to correct the previous one
- The “wrong” changeset will hit your production database followed by the “right” changeset

Tips and tricks - stored procedures

If you have stored procedures the should obviously be versioned controlled

But the CRC does not fit well as that requires the stored procedure to be repeated for every modification

The “run always” attribute saves the day. This will make sure that whenever there is a change it is applied -- a better fit for stored procedures

Tips and tricks - stored procedures

```
--liquibase formatted sql
--changeset rikard:1 runOnChange:true

ALTER procedure jfokus
...
```

- Above we use a plain sql file with two special comments that magically makes it a changeset

Major Concepts - Contexts

Contexts can be applied to changesets to control which are ran in different environments

For example, some changesets can be tagged as "production" and others as "test"

Use Contexts only when there is a good reason

Tips and Tricks - neutralizing data

- The “run always” attribute can also be used to neutralize data in dev and test environments
- Useful in cases when the database is copied for production to other environments
- The changes that has been applied is stored in the database itself, therefore it is safe to copy clone a database

Tips and Tricks - neutralizing data

```
<changeSet id="1" author="joe" runAlways="true" context="sqld,sqlt,localhost">
    <comment>Change users psw on (localhost, dev, test) to 'Secret' and their email to 'test@jfokus.se'</comment>
    <update tableName="user">
      <column name="email" value="test@jfokus.se"/>
      <column name="password" value="Secret"/>
      <where>email != 'test@jforum.se' OR password != 'Secret'</where>
    </update>
</changeSet>
```

Alternatives

- Flyway
- c5-db-migration
- dbdeploy
- mybatis
- MIGRATEdb
- migrate4j
- dbmaintain
- AutoPatch

Real life experience

We have used Liquibase in projects for many years, more than 50 man years of development with very little problems

Overall score 97%

by Rikard