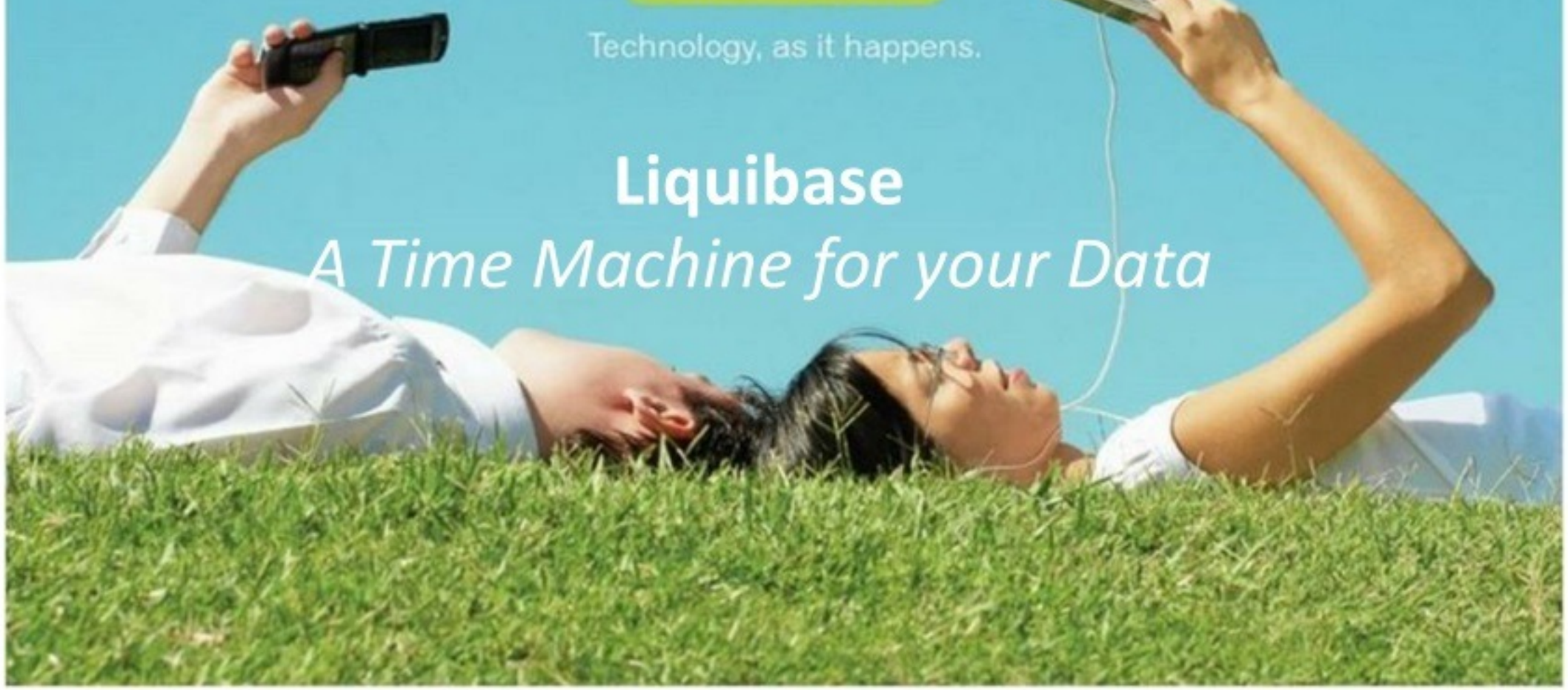
Architecting Applications for Mobile and Cloud

neev TECHNOLOGIES

Technology, as it happens.



Liquibase

A Time Machine for your Data

neev TECHNOLOGIES

Web . Mobile . Cloud

About Neev

Web

Magento eCommerce
SaaS Applications
Video Streaming Portals
Rich Internet Apps
Custom Development

Mobile

iPhone
Android
Windows Phone 7
HTML5 Apps

Cloud

AWS Consulting Partner
Rackspace
Joyent
Heroku
Google App Engine

Key Company Highlights

250+ team with experience in managing offshore, distributed development.

Neev Technologies established in Jan '05

VC Funding in 2009 By Basil Partners

Part of Publicis Groupe

Member of NASSCOM.

Development Centers in Bangalore and Pune.

Offices at Bangalore, USA, Delhi, Pune, Singapore and Stockholm.

User Interface Design and User Experience Design

Performance Consulting Practices

Quality Assurance & Testing

Outsourced Product Development

neev TECHNOLOGIES
Web . Mobile . Cloud

Why DB Versioning?

• Frequent revision is at the heart of most of the agile projects. We, at Neev, deal with requirement revisions, application code changes, database schema revisions across the sprints. In a word everything undergoes change.

• Though it is easy to understand that the code and application would change as revisions occur, developers generally are not comfortable in handling DB versions.

• Any time a build needs to be reverted to a previous version, DB schema rollback also needs to be done, which without a professional tool is not very straight forward.

• A source control tool for DB schema is as necessary as tools for source code control like GIT.

• You may argue that why don't we keep the schema definition as SQL scripts in GIT.

• This may help in recreating the DB but when it comes to dropping just a column or changing the datatype, entire DDL scripts may be too expensive in terms of data restoration and time taken.

•We found Database (DB) versioning is mandatory as code versioning.

Why Liquibase?

- We explored few tools in this space, like
  - dbdeploy
  - dbmaintain
  - Liquibase
- Liquibase stands out from rest of the lot in terms of
  - Ease of use
  - Learning curve
  - Plugin support
  - DB support

Features of Liquibase

Some feature claimed by Liquibase are:

1. All changes to the database are stored in XML/JSon/Yaml or SQL files and identified by a combination of an “ID” and “author” tag as well as the name of the file itself.
2. Updates database to current version
3. Rollback last X changes to database, date/time by tag
4. SQL for Database updates and Rollbacks can be saved for manual review
5. Stand-alone IDE and Eclipse plug-in
6. Database diff report, diff changelog generation
7. Ability to create changelog to generate an existing database
8. Database change documentation generation
9. DBMS Check, user check, and SQL check preconditions
10. Executable via command line, Ant, Maven, Servlet container, or Spring
11. Support for more than 10 RDBMS
12. Grails DB migration plugin uses Liquibase.

Database Change Log File

- The root of all Liquibase changes is the databaseChangeLog file.
- The Sample Change Log File is as given below:

```
<databaseChangeLog>
<changeSet id="1" author="sougata">
<createTable tableName="event">
<column name="id" type="int">
<constraints primaryKey="true" nullable="false"/>
</column>
<column name="name" type="varchar(50)">
<constraints nullable="false"/>
</column>
<column name="active" type="boolean" defaultValueBoolean="true"/>
</createTable>
</changeSet>
<changeSet author="sougata" id="tagDatabase-example">
<tagDatabase tag="0.1"/>
</changeSet>
</databaseChangeLog>
```

Rollback

• Specifying a tag to rollback will rollback all change-sets that were executed against the target database after the given tag was applied.

• You can specify the number of change-sets to rollback.

• You can specify the date to roll back to.

```
<changeSet id="changeRollback" author="sougata">
<createTable tableName="event">
<column name="id" type="int"/>
</createTable>
<rollback>
<dropTable tableName="event"/>
</rollback>
</changeSet>
```

Generating ChangeLog File

- For generating change log file from existing database, we used command line tool.

- Here is a example of the Liquibase command:

./liquibase –driver=com.mysql.jdbc.Driver –classpath=$DB_JAR/mysql-connector-java-5.1.21.jar –changeLogFile=app-changelog/generated-changeLog.xml –url=”jdbc:mysql://localhost/myapp” –username=root –password=secret generateChangeLog

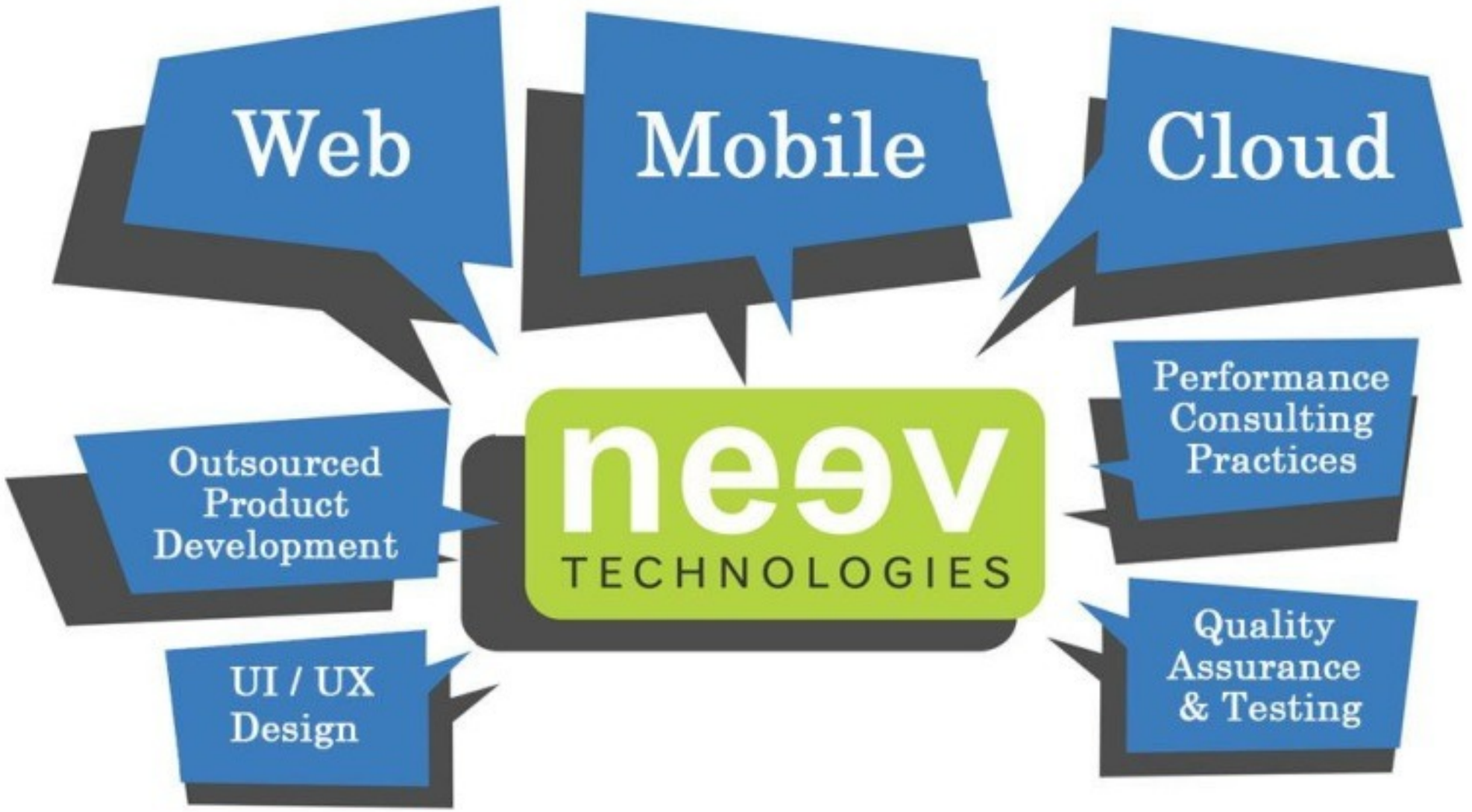
Best Practices

Some of the best practices we follow as mentioned below:

- Organize Change Log File
  - We organize our application version files in seperate changeLog file and we have a master ChangeLog File.
  - Below is the masterChangeLog file.

```
<?xml version="1.0" encoding="UTF-8"?>
<databaseChangeLog>
<include file="com/myapp/db/changelog/db.changelog-1.0.0.xml"/>
<include file="com/myapp/db/changelog/db.changelog-1.0.1.xml"/>
<include file="com/myapp/db/changelog/db.changelog-1.0.2.xml"/>
</databaseChangeLog>
```

- Change Set ID
  - Provide change set ID for all the changes.
- Use Liquibase from the initial stage of the development.



Neev Information Technologies Pvt. Ltd.

sales@neevtech.com

India - Bangalore

The Estate, # 121,6th Floor,
Dickenson Road
Bangalore-560042
Phone :+91 80 25594416

India - Pune

#13 L'Square, 3rd Floor
Parihar Chowk, Aundh,
Pune – 411007.
Phone : +91-64103338

USA

1121 Boyce Rd Ste 1400,
Pittsburgh PA 15241
Phone : +1 888-979-7860

Sweden

Neev AB, Birger Jarlsgatan
53, 6tr,
11145, Stockholm
Phone: +46723250723

Singapore

#08-03 SGX Centre 2, 4
Shenton Way,
Singapore 068807
Phone: +65 6435 1961

Visit us at Neevtech.com to know more about our offerings.

neev TECHNOLOGIES
Web . Mobile . Cloud