Many organizations have implemented DevOps in their applications, that's true. But, at the same time, their database change process hasn't realized any of these benefits and is still left in the dark ages. But what if you could automate that too? Yeah, you guessed right – it can be done using Liquibase. And here's a Liquibase tutorial to show you how to do that.

Is this Liquibase tutorial for you?

Are you manually executing scripts to your database? Or maybe you're wasting time validating database scripts received from your team?

After that, are you merging scripts into one file and executing them in every environment? How about deployment errors? Have you ever spent hours looking at who, why, and what was changed in the database?

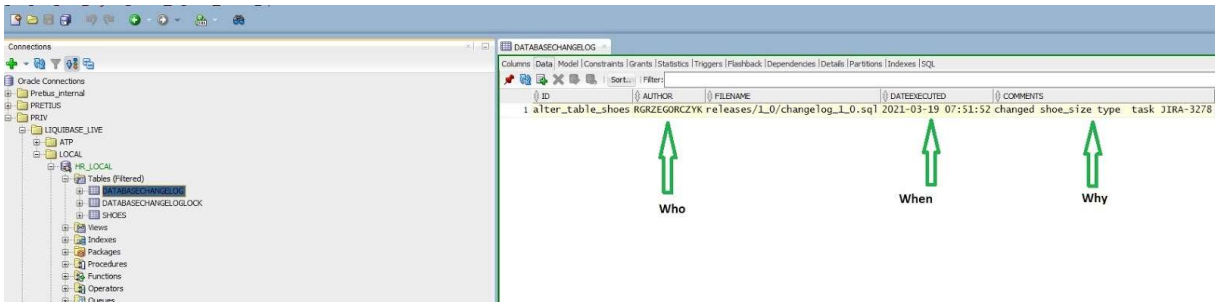
But what if you can't have an entire CI/CD process right now or company policy doesn't allow you to run scripts on specific environments? That's not a problem for Liquibase.

By using Liquibase you can:

- automate your database deployment scripts,
- consistently deploy the same way in every environment,
- have rollbacks always prepared for every database change,
- have all detailed info of deployments in one place.

What's more, thanks to this you will have:

- fewer deployment errors,
- happy and efficient developers coding together on the same databases,
- every change audited, e.g who, when (and why) changed the column SHOES.SHOE_SIZE from a NUMBER data type to a VARCHAR2,
- more coffee time.



Wanna know who, when and why changed your database column? Keep on reading this Liquibase tutorial

In a series of articles, I'll show you how we automated our database change process at Pretius using Liquibase and GIT – examples from limited-access environments included. Let's start with this basic Liquibase tutorial.

What is Liquibase exactly?

Liquibase (LB) is an open source tool written in Java. It makes defining database changes easy, in a format that's familiar and comfortable to each user. Then, it automatically generates database-specific SQL for you.

Database changes (every change is called changeset) are managed in files called changelogs.

Liquibase needs two tables at your db schema(created automatically):

- DATABASECHANGELOG — a table storing information about all changes made to your database,
- DATABASECHANGELOGLOCK — used to prevent users from doing changes to the database at the same time.

My examples will be based on changesets written in SQL — it's the easiest way to start automating your Oracle database change process.

Start with installing Liquibase

Go to https://www.liquibase.org/download and download the latest version — choose “Just the files”. In this article, I will use version 4.3.0. built 09.02.2021.

Extract the downloaded zip folder (e.g., to disk C:). After that, you must set a New Path System Variable to the liquibase-version#bin folder on your computer. For Liquibase to work properly, you must also have JAVA installed.

Go to your favourite CLI tool (I use Visual Studio Code) and type:

```
liquibase —version
```

If everything’s ok, you will see:

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR> liquibase --version
##################################################
##                                              ##
##  Get documentation at docs.liquibase.com     ##
##  Get certified courses at learn.liquibase.com ##
##  Free schema change activity reports at      ##
##      https://hub.liquibase.com               ##
##                                              ##
##################################################
Starting Liquibase at 18:05:00 (version 4.3.0 #16 built at 2021-02-09 15:47+0000)
Liquibase Version: 4.3.0
Liquibase Community 4.3.0 by Datical
Running Java under C:\Program Files\Java\jdk1.8.0_241\jre (Version 1.8.0_241)
PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR>
```

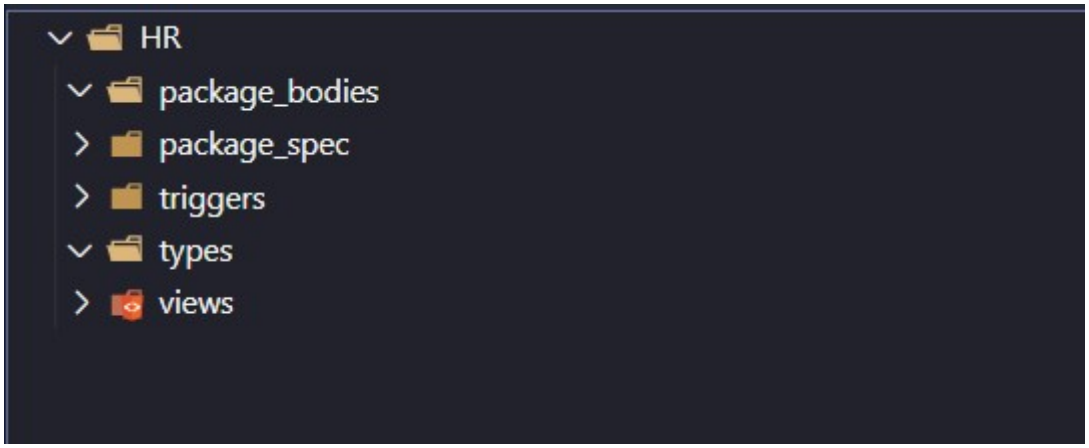
If you use UTF8 encoding in your files remember to edit the **liquibase.bat** file by adding line:

```
IF NOT DEFINED JAVA_OPTS set JAVA_OPTS=-Dfile.encoding=UTF-8
```

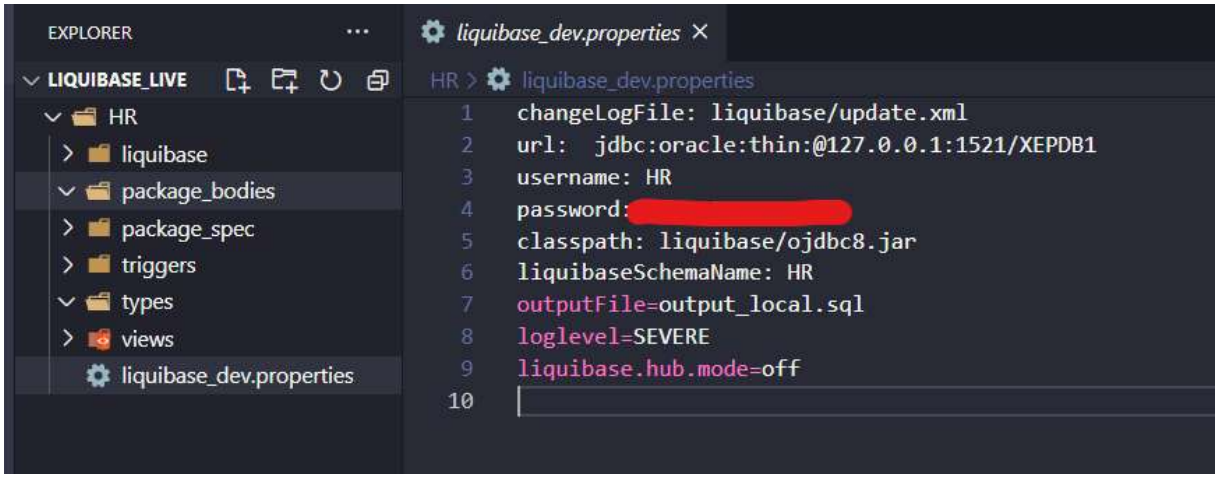
```
liquibase.bat
26  set CP=!CP!;!LIQUIBASE_HOME!lib
27
28  rem special characters may be lost
29  setlocal DISABLEDELAYEDEXPANSION
30
31  IF NOT DEFINED JAVA_OPTS set JAVA_OPTS=-Dfile.encoding=UTF-8
32
33  set JAVA_PATH=java
34  if NOT "%JAVA_HOME%" == "" set JAVA_PATH=%JAVA_HOME%\bin\java
```

Configure your project and Liquibase

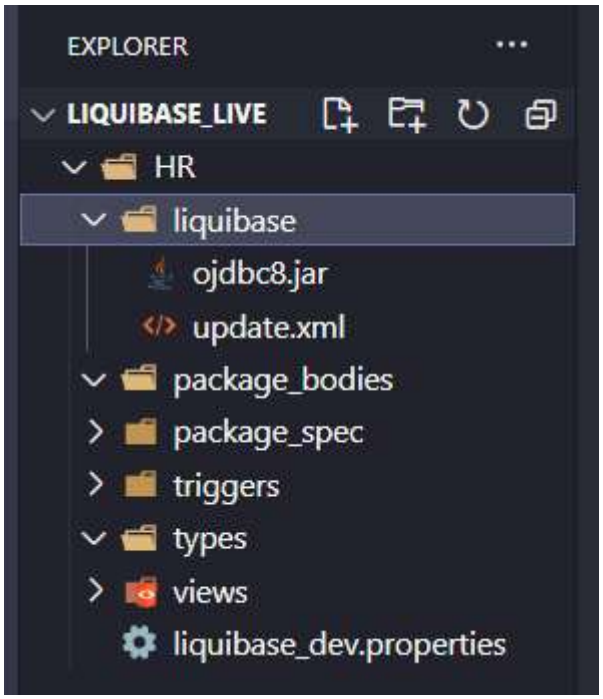
Ok, let's see how we can organize our files (folder HR is my GIT repository). In these folders, we will put files created during project development. If you had other types of objects (which are "create or replace" type) just create folder with it, e.g "synonyms".



Now, we need to create Liquibase properties file with connection to our DEV database:



```
#path to our master changelog file
changeLogFile: liquibase/update.xml
#dbhost and credentials
url: jdbc:oracle:thin:@127.0.0.1:1521/XEPDB1
username: HR
password: XXXXXX
#OJDBC driver localization
classpath: liquibase/ojdbc8.jar
#schema, where Liquibase will store it’s DATABASECHANGELOG and
DATABASECHANGELOGLOCK table (if other than HR, remember to add grants to HR!)
liquibaseSchemaName: HR
#default SQL file name generated by Liquibase
outputFile=output_local.sql
#debug mode
loglevel=SEVERE
#extra option from Liquibase, we don’t need it for now.
liquibase.hub.mode=off
```



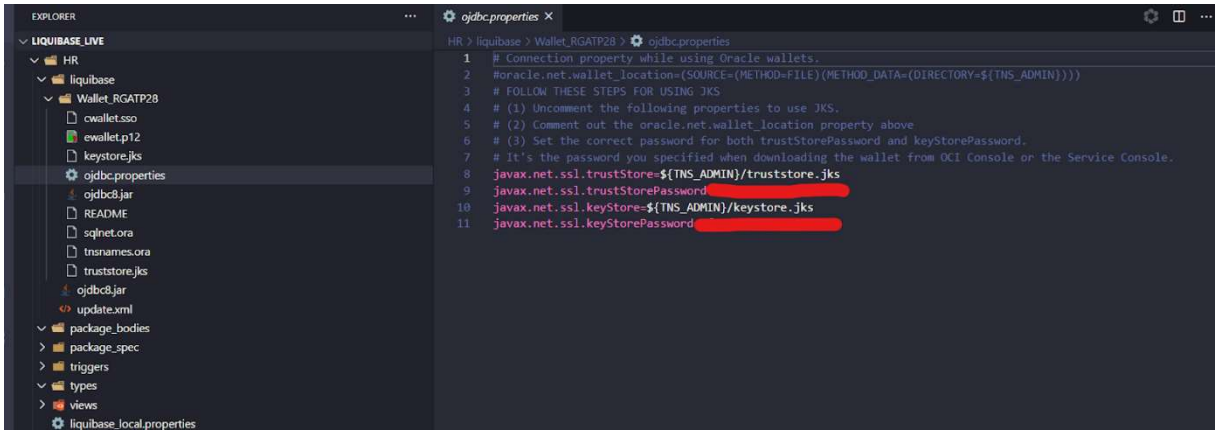
Now, create an **update.xml** file (put it into new hr/liquibase folder with ojdbc file):

```
<?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"
xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-
4.3.xsd"></databaseChangeLog>
```

Use Oracle Wallet (optional)

If your Oracle database is hosted on Oracle Autonomous Database, you need to use the wallet to connect to it through Liquibase. Therefore, download your wallet and remember the password for it.

Unpack your **WALLET_NAME.ZIP** to previously created HR/liquibase folder. Also edit your **HR/liquibase/wallet_name/ojdbc.properties** file:



Your file should look like on the screen above. In the lines **javax.net.ssl.trustStorePassword** and **javax.net.ssl.keyStorePassword**, put your ATP wallet password.

Edit URL at your **liquibase_local.properties** file and set your connection name (from **Wallet/tnsnames.ora** and path to wallet):

```
url:  jdbc:oracle:thin:@rgatp28_high?TNS_ADMIN=liquibase/Wallet_RGATP2
8
```

Check your **sqlnet.ora** file, make sure there is "SSL_SERVER_DN_MATCH=yes". Don't change anything else.

Connect Liquibase with a database

If everything is set properly, we can make the first connection to our DEV database. Start your favourite CLI from the HR folder (location of liquibase properties file) – for the purpose of this article, I use terminal directly from VS Code and connection to my local development database.

```
liquibase —defaultsFile=liquibase_dev.properties updateSQL
```

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE                                  2: powershell

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR> liquibase --defaultsFile=liquibase_dev.properties updateSQL
Liquibase Community 4.3.0 by Datical
####################################################
```

liquibase -> invocation of LB(environment path)

defaultsFile -> name and location of our properties file

(if you'd name properties file to "liquibase.properties" then you may omit this command because it's liquibase default. I'll prefer to have different names for every connection)

updateSQL -> Liquibase command, only generation of SQL script (it won't do anything on your database)

In a few second LB will generate **output_file.sql**

```
10  -- Create Database Lock Table
11  CREATE TABLE HR.DATABASECHANGELOGLOCK (ID INTEGER NOT NULL, LOCKED NUMBER(1) NOT NULL, LOCKGRANTED TIMESTAMP, LOCKEDBY VARCHAR2(255), CONSTRAINT PK_DATABASECHANGELOGLOCK PRIMAR
12
13  -- Initialize Database Lock Table
14  DELETE FROM HR.DATABASECHANGELOGLOCK;
15
16  INSERT INTO HR.DATABASECHANGELOGLOCK (ID, LOCKED) VALUES (1, 0);
17
18  -- Lock Database
19  UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 1, LOCKEDBY = 'DESKTOP-H61VFP7 (192.168.8.102)', LOCKGRANTED = TO_TIMESTAMP('2021-03-22 12:58:29.119', 'YYYY-MM-DD HH24:MI:SS.FF')
20
21  -- Create Database Change Log Table
22  CREATE TABLE HR.DATABASECHANGELOG (ID VARCHAR2(255) NOT NULL, AUTHOR VARCHAR2(255) NOT NULL, FILENAME VARCHAR2(255) NOT NULL, DATEEXECUTED TIMESTAMP NOT NULL, ORDEREXECUTED INT
23
24
25  -- Release Database Lock
26  UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 0, LOCKEDBY = NULL, LOCKGRANTED = NULL WHERE ID = 1;
27
28
```

As you can see, if you'd run this script to your database it would create two tables: DATABASECHANGELOG and DATABASECHANGELOGLOCK.

Ok, now let's create those tables:

```
liquibase –defaultsFile=liquibase_dev.properties update
```

Update command will run execute SQL to database.

Tables are created:

Now, we need to create a changelog file that will point to our folders with objects (those we can create / replace).

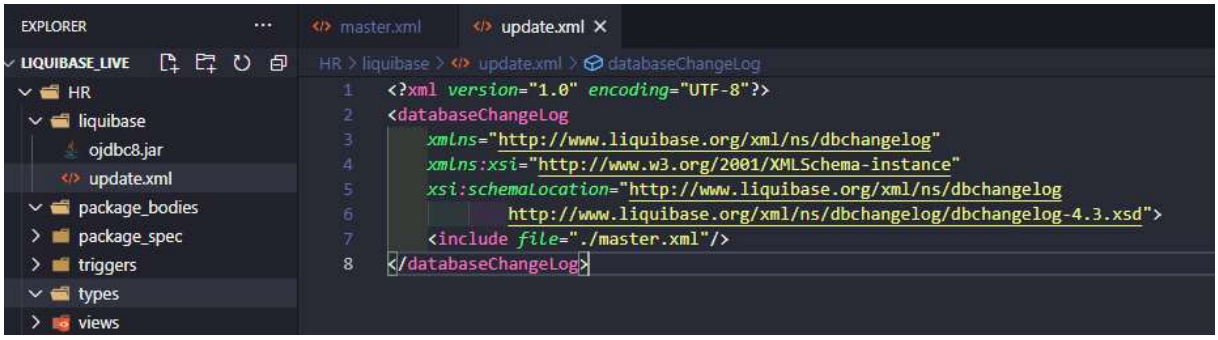
I created **HR/master.xml** file:

```
<?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"
xsi:schemaLocation="http://www.liquibase.org/xml/ns/dbchangelog
http://www.liquibase.org/xml/ns/dbchangelog/dbchangelog-4.3.xsd">
<includeAll path="triggers" relativeToChangelogFile="true"/>
<includeAll path="views" relativeToChangelogFile="true"/>
<includeAll path="types" relativeToChangelogFile="true"/>
<includeAll path="package_spec" relativeToChangelogFile="true"/>
<includeAll path="package_bodies" relativeToChangelogFile="true"/>
</databaseChangeLog>
```

It points to my objects folders and all of it's content.

In main changelog **HRliquibaseupdate.xml** set path to your **master.xml** file, just add line:

```
<include file="./master.xml"/>
```



Liquibase always runs from our liquibase_dev.properties file and update.xml file, so It must see all of your files from there.

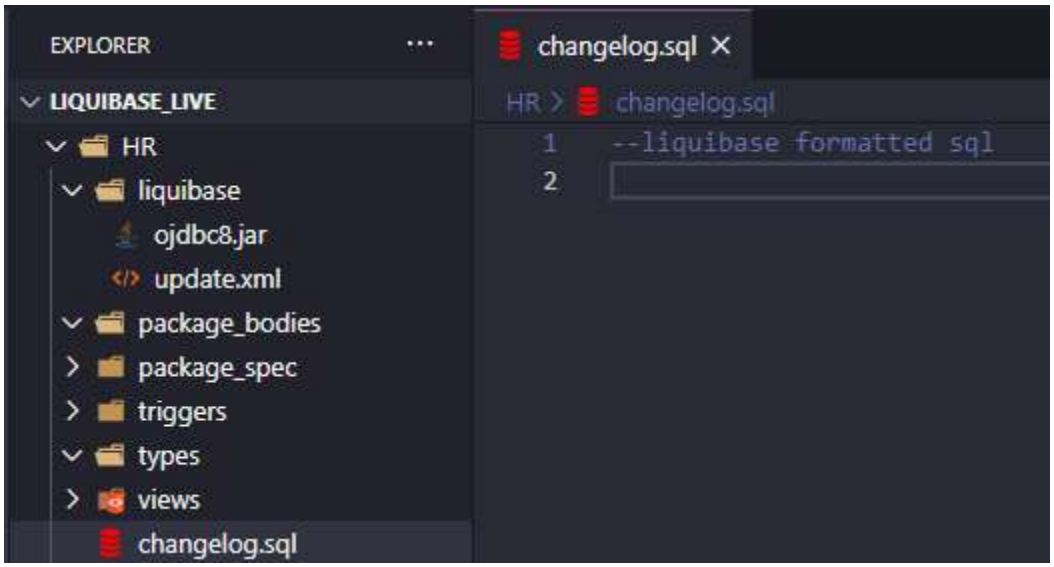
Track your DML and DDL database changes

Ok, wait… but what about changes like DDL or DML? No problem.

For that type of change we create a separate changelog file and write our changesets inside of it.

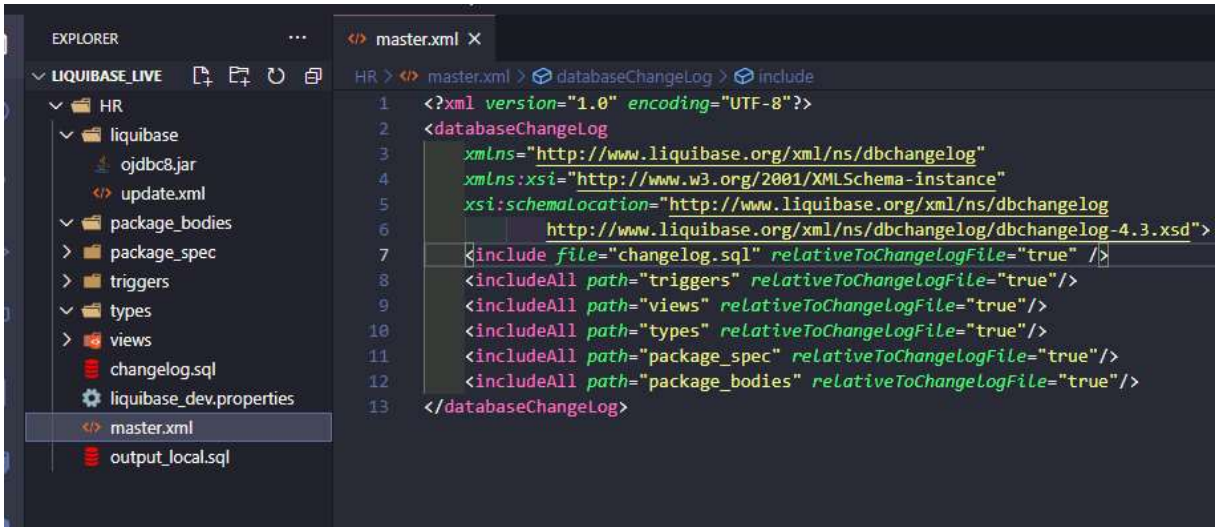
Just create **changelog.sql** file and mark it as Liquibase sql file by typing:

```
—liquibase formatted sql
```



Point to our new changelog in **master.xml** file by adding:

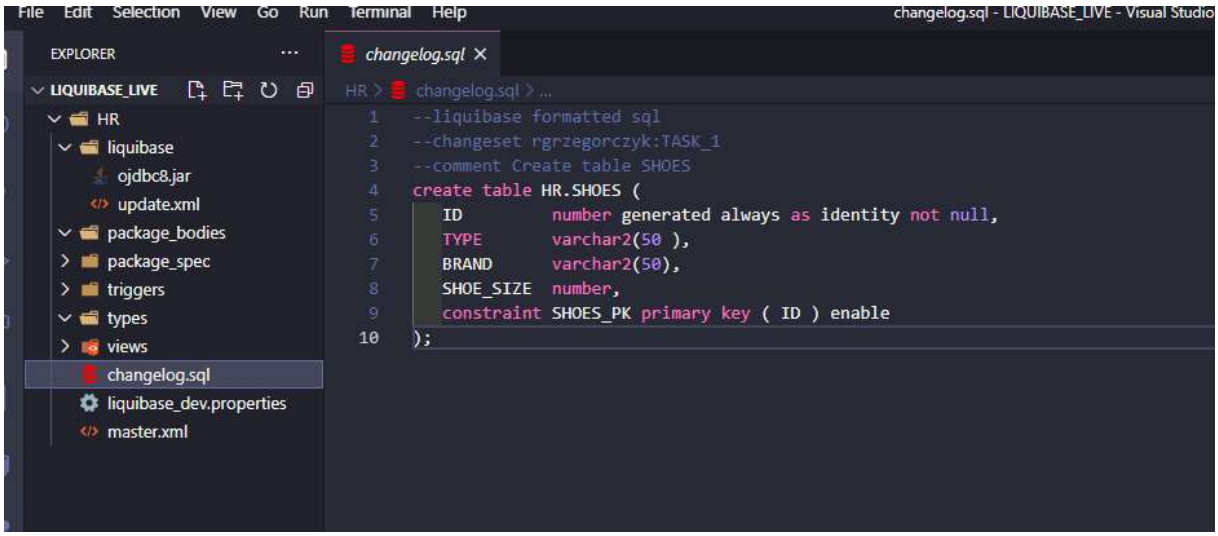
<include file="changelog.sql" relativeToChangelogFile="true" />



Order in which you point to your changelogs or folders is very important. It tells Liquibase in which order to run your sql. It is better to run changelogs first (inside of which is "create table(...)") and after that compile package which uses this table.

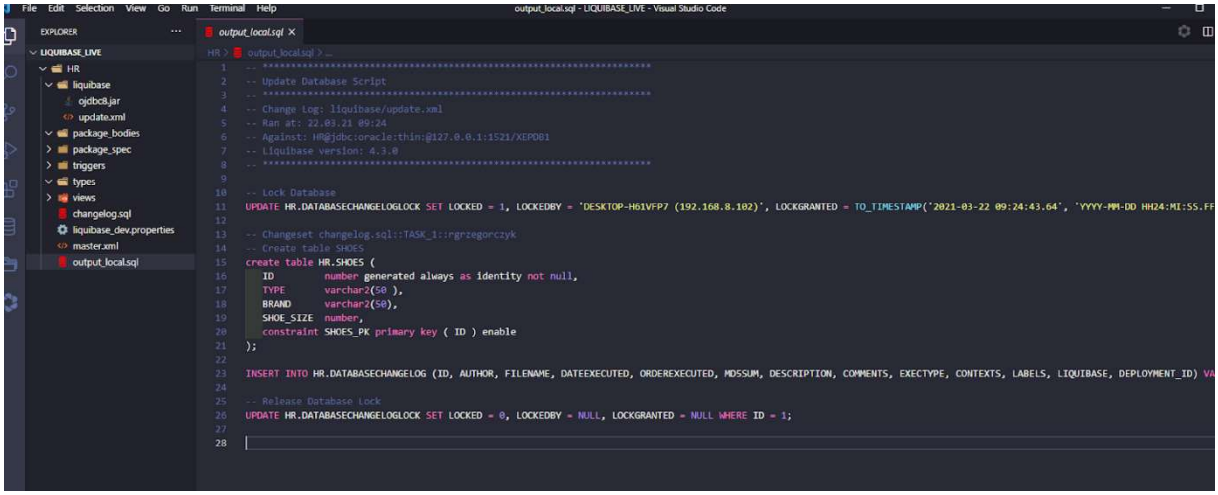
Let's create first project table in our changeset. Just write:

```
—changeset AUTHOR:CHANGESET_NAME
—comment OPTIONAL COMMENT
YOUR DDL
```



Let's ask LB to generate our SQL file (just to preview what changes are going to be made to our database).

```
liquibase –defaultsFile=liquibase_dev.properties updateSQL
```



As you may noticed, LB is going to lock our DATABASECHANGELOGLOCK table by setting LOCKED = 1 (while you are running your script to DB, column LOCKED is set to 1. When another user runs LB in the same time, Liquibase will wait until lock is released), then it will create a SHOES table, insert log change into DATABASECHANGELOG and release lock from DATABASECHANGELOGLOCK table.

If everything is fine, execute script to our database:

```
liquibase –defaultsFile=liquibase_dev.properties update
```

The table SHOES has been created.

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE

Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR> liquibase --defaultsFile=liquibase_dev.properties update
Liquibase Community 4.3.0 by Datical
##################################################
##                                              ##
##  /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \    ##
## ( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \    ##
##  \\/  ___)| |_)| | | | | || (_| |  ) ) ) )    ##
##   '  |____| .__|_| |_|_| |_\__, | / / / /    ##
##  =========|_|==============|___/=/_/_/_/      ##
##                                              ##
##  Get documentation at docs.liquibase.com     ##
##  Get certified courses at learn.liquibase.com ##
##  Free schema change activity reports at      ##
##      https://hub.liquibase.com               ##
##                                              ##
##################################################
Starting Liquibase at 09:39:45 (version 4.3.0 #16 built at 2021-02-09 15:47+0000)
Liquibase: Update has been successful.
PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR>
```

We can check who, why and when created this table.

Finally, nothing is anonymous!

DATABASECHANGELOG | HR_LOCAL

Worksheet | Query Builder

```
select author,
       id,
       filename,
       dateexecuted,
       comments
  from databasechangelog;
```

Query Result

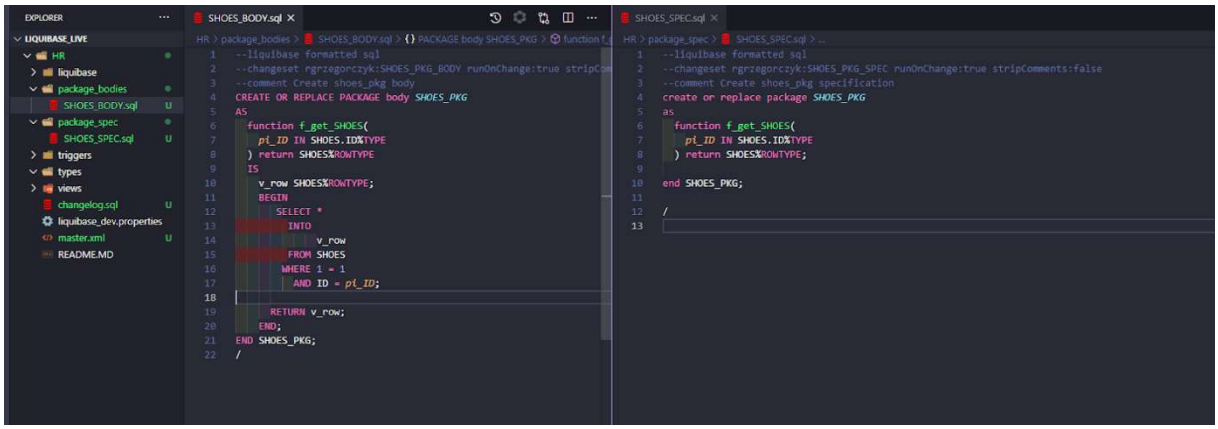
SQL | All Rows Fetched: 1 in 0,003 seconds

	AUTHOR	ID	FILENAME	DATEEXECUTED	COMMENTS
1	rgrzegorczyk	TASK_1	changelog.sql	2021-03-22 09:39:47	Create table SHOES

Track other database changes (packages, views, etc.)

Now, we can do the same with some scripts. I created a package called SHOES_PKG in 2 separate files. Every file is unique changeset and should be marked as liquibase formatted sql file.

Sql file is unique changeset with additional parameters:



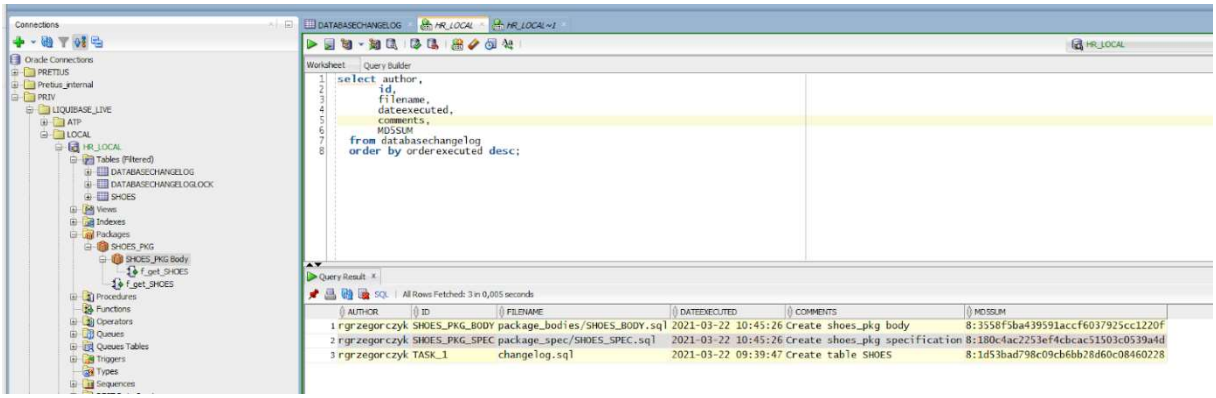
runOnChange:true — it means, everytime we change our package Liquibase will run this changeset against our database (compile this package)

stripComments:false — do not cut our code comments

Now, if we check what SQL would LB run against database (updateSQL) — it would compile both package spec and package body.

Let's compile these package in our DB (update command).

Everything is logged and packages are compiled.



Have a look at MD5SUM column value — it's last checksum of your changeset.

For now, all pending changes are executed, LB will not generate anything in SQL (besides locking LB table) — check it by running **updateSQL**.

```
-- *******************************************************************
-- Update Database Script
-- *******************************************************************
-- Change Log: liquibase/update.xml
-- Ran at: 22.03.21 10:48
-- Against: HR@jdbc:oracle:thin:@127.0.0.1:1521/XEPDB1
-- Liquibase version: 4.3.0
-- *******************************************************************

-- Lock Database
UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 1, LOCKEDBY = 'DESKTOP-H61VFP7 (192.168.8.102)', LOCKGRANTED = TO_TIMESTAMP('2021-03-22 10:48:21.541', 'YYYY-MM-DD HH24:MI:SS.FF

-- Release Database Lock
UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 0, LOCKEDBY = NULL, LOCKGRANTED = NULL WHERE ID = 1;
```

Now, let’s change our SHOES_PKG body and save the file.

```
--liquibase formatted sql
--changeset rgrzegorczyk:SHOES_PKG_BODY runOnChange:true stripComments:false
--comment Create shoes_pkg body
CREATE OR REPLACE PACKAGE body SHOES_PKG
AS
  function f_get_SHOES(
    pi_ID IN SHOES.ID%TYPE
  ) return SHOES%ROWTYPE
  IS
    v_row SHOES%ROWTYPE;
    BEGIN
      SELECT *
        INTO
            v_row
        FROM SHOES
       WHERE 1 = 1
         AND ID = pi_ID;

      RETURN v_row;
    END;
END SHOES_PKG;
/
```

```
--liquibase formatted sql
--changeset rgrzegorczyk:SHOES_PKG_BODY runOnChange:true stripComments:false
--comment Create shoes_pkg body
CREATE OR REPLACE PACKAGE body SHOES_PKG
AS
  function f_get_SHOES(
    pi_ID IN SHOES.ID%TYPE
  ) return SHOES%ROWTYPE
  IS
    v_row SHOES%ROWTYPE;
    BEGIN
      SELECT *
        INTO
            v_row
        FROM SHOES
       WHERE 1 = 1
         AND ID = pi_ID;
+-- test comment

      RETURN v_row;
    END;
END SHOES_PKG;
/
```

Checksum of the file has changed and LB will compile this package again — let’s run an update.

EXPLORER

LIQUIBASE_LIVE

- HR
  - liquibase
  - package_bodies
    - SHOES_BODY.sql M
  - package_spec
  - triggers
  - types
  - views
  - changelog.sql
  - liquibase_dev.properties
  - master.xml
  - README.MD

SHOES_BODY.sql

HR > package_bodies > SHOES_BODY.sql > { } PACKAGE body SHOES_PKG > function f_get_SHOES

```
--liquibase formatted sql
--changeset rgrzegorczyk:SHOES_PKG_BODY runOnChange:true stripComments:false
--comment Create shoes_pkg body
CREATE OR REPLACE PACKAGE body SHOES_PKG
AS
  function f_get_SHOES(
    pi_ID IN SHOES.ID%TYPE
  ) return SHOES%ROWTYPE
  IS
    v_row SHOES%ROWTYPE;
    BEGIN
      SELECT *
        INTO
              v_row
        FROM SHOES
       WHERE 1 = 1
         AND ID = pi_ID;
-- test comment

      RETURN v_row;
    END;
END SHOES_PKG;
/
```

TERMINAL PROBLEMS OUTPUT DEBUG CONSOLE

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR> liquibase --defaultsFile=liquibase_dev.properties update
Liquibase Community 4.3.0 by Datical
####################################################
##   _   _                _ _                      ##
##  | |  (_)              (_) |                    ##
##  | |    _  __ _  _ _   _| |__   __ _ __  ___    ##
##  | |   | |/  _` | | | | | '_ \ / _` / __|/ _ \  ##
##  | |___| | (_| | |_| | | |_) | (_| \__ \  __/  ##
##  \_____/_|\__, |\__,_|_|_.__/ \__,_|___/\___|   ##
##            | |                                   ##
##            |_|                                   ##
##                                                  ##
##  Get documentation at docs.liquibase.com         ##
##  Get certified courses at learn.liquibase.com    ##
##  Free schema change activity reports at          ##
##      https://hub.liquibase.com                   ##
##                                                  ##
####################################################
Starting Liquibase at 13:06:48 (version 4.3.0 #16 built at 2021-02-09 15:47+0000)
Liquibase: Update has been successful.
PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR>
```

Connections

Oracle Connections
- PRETIUS
- Pretius_internal
- PRIV
  - LIQUIBASE_LIVE
    - ATP
    - LOCAL
      - HR_LOCAL
        - Tables (Filtered)
          - DATABASECHANGELOG
          - DATABASECHANGELOGLOCK
          - SHOES
        - Views
        - Indexes
        - Packages
          - SHOES_PKG
            - SHOES_PKG Body
              - f_get_SHOES
        - Procedures
        - Functions
        - Operators
        - Queues
        - Queues Tables
        - Triggers
        - Types
        - Sequences
        - REST Data Services
        - Materialized Views
        - Materialized View Logs
        - Synonyms
        - Public Synonyms
        - Database Links
        - Public Database Links
        - Directories
        - Editions
        - Application Express
        - Java
        - XML Schemas
        - XML DB Repository
        - OLAP Option
        - Analytic Views
        - Scheduler
        - RDF Semantic Graph

DATABASECHANGELOG | HR_LOCAL | HR_LOCAL~1

Worksheet | Query Builder

```
select author,
       id,
       filename,
       dateexecuted,
       comments,
       MD5SUM
  from databasechangelog
  order by orderexecuted desc;
```

Query Result — All Rows Fetched: 3 in 0,001 seconds

	AUTHOR	ID	FILENAME	DATEEXECUTED	COMMENTS	MD5SUM
1	rgrzegorczyk	SHOES_PKG_BODY	package_bodies/SHOES_BODY.sq	2021-03-22 10:59:1	Create shoes_pkg body	8:b1fd8bca59b3baea86e4782d65e61f01
2	rgrzegorczyk	SHOES_PKG_SPEC	package_spec/SHOES_SPEC.sql	2021-03-22 10:45:26	Create shoes_pkg specification	8:180c4ac2253e14cbca6 [illegible]
3	rgrzegorczyk	TASK_1	changelog.sql	2021-03-22 09:39:47	Create table SHOES	8:1d53bad798c09cb6bb28d60c08460228

HR_LOCAL

Worksheet | Query Builder

```
select author,
       id,
       filename,
       dateexecuted,
       comments,
       MD5SUM
  from databasechangelog
  order by orderexecuted desc;
```

Query Result — All Rows Fetched: 3 in 0,002 seconds

	AUTHOR	ID	FILENAME	DATEEXECUTED	COMMENTS	MD5SUM
1	rgrzegorczyk	SHOES_PKG_BODY	package_bodies/SHOES_BODY.sq	2021-03-22 10:45:26	Create shoes_pkg body	8:3558f5ba439591accf6037925cc1220f
2	rgrzegorczyk	SHOES_PKG_SPEC	package_spec/SHOES_SPEC.sql	2021-03-22 10:45:26	Create shoes_pkg specification	8:180c4ac2253e14cbca6 [illegible]
3	rgrzegorczyk	TASK_1	changelog.sql	2021-03-22 09:39:47	Create table SHOES	8:1d53bad798c09cb6bb28d60c08460228

SQL History | Dbms Output | Messages - Log

Liquibase compiled package body again and updated row with these changeset in DATABASECHANGELOG table – with actual DATEEXECUTED and new MD5SUM value.

How to install Liquibase in an existing software project?

In this part of the Liquibase tutorial, you will learn how to implement database automation in an existing software project.

Is it possible without hours of additional work? Yes!

There are a few ways to automate your existing database using Liquibase. I will show you two which I found most useful – and you can choose the one that suits your needs best.

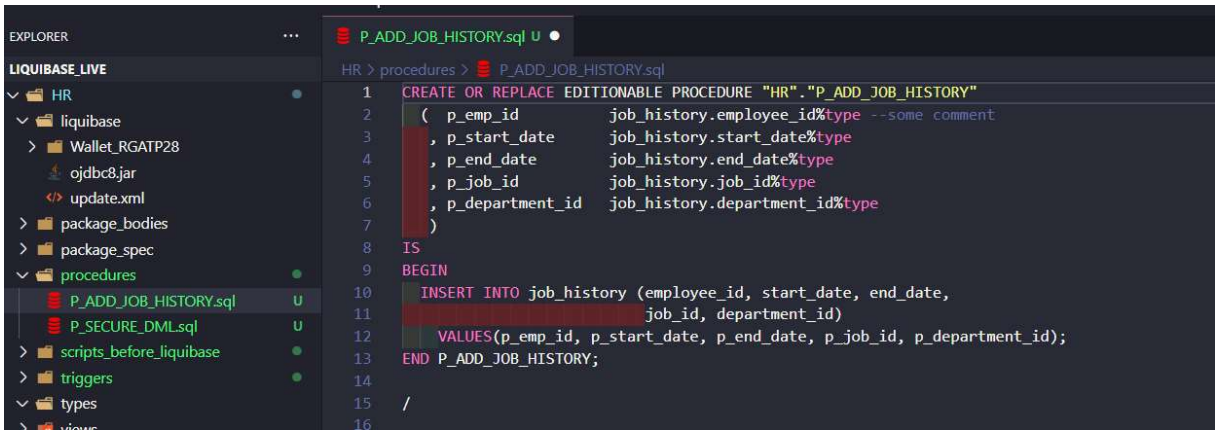
In the examples below, I'll be using the project created in the previous steps of this Liquibase tutorial.

How to install Liquibase when there are lots of objects in your existing project

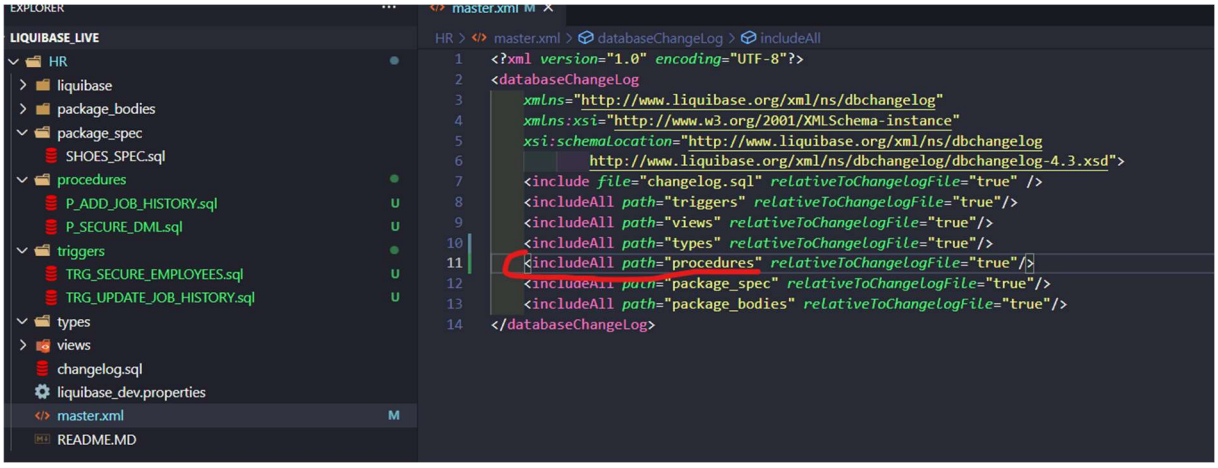
Configure Liquibase in your project repository and leave all files as they are – just remember to add a path to them in your **master.xml** file.

So, I have created 2 procedures and 2 triggers before implementing Liquibase:

```
P_ADD_JOB_HISTORY
P_SECURE_DML
TRG_SECURE_EMPLOYEES
TRG_UPDATE_JOB_HISTORY
```



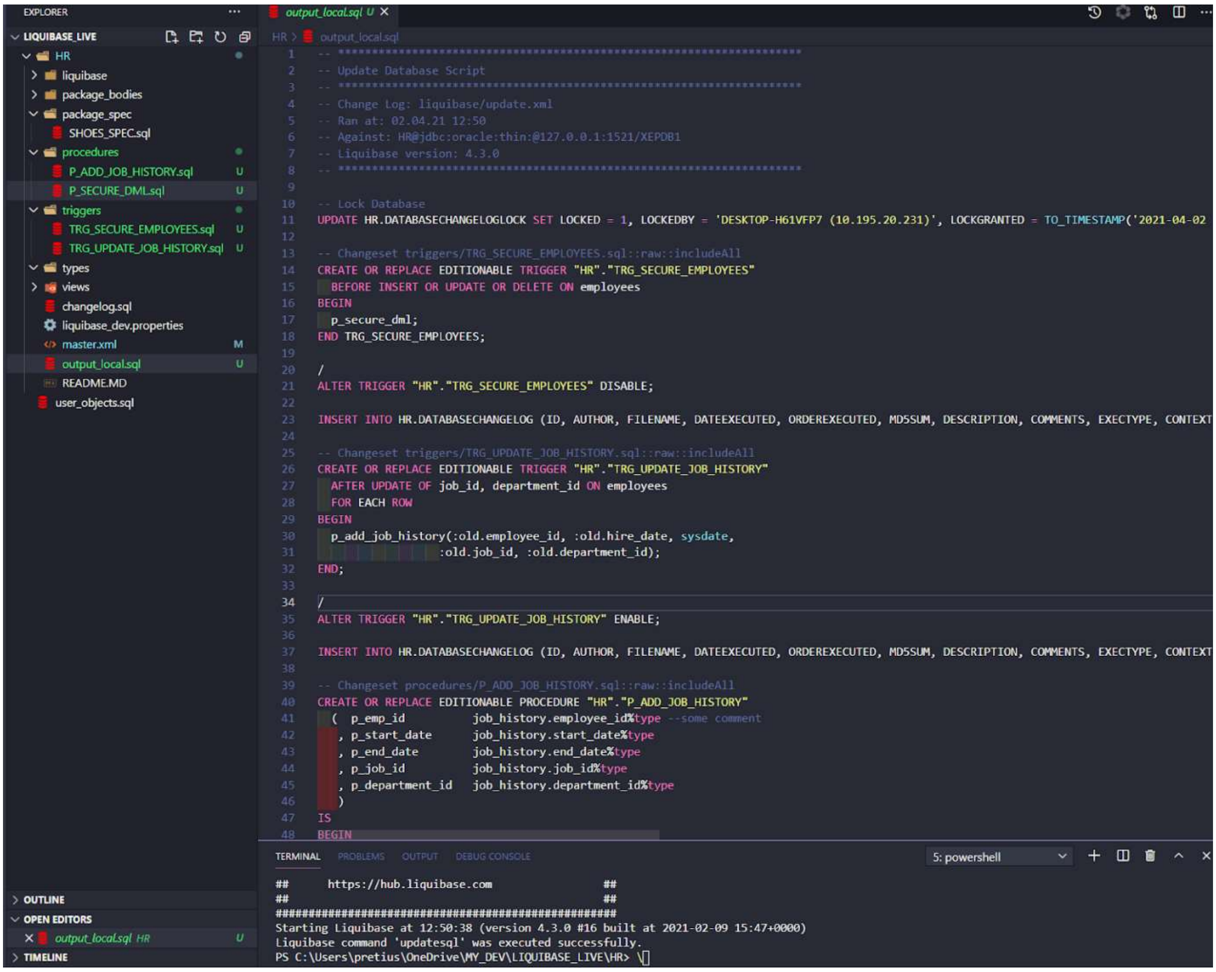
You DON’T need to add “changeset” or “–liquibase formatted sql” to your file right now.



I also added a path to a **PROCEDURES** folder to my **master.xml**.

Now, let’s run Liquibase **updateSQL** to see what SQL Liquibase would like to execute:

```
liquibase –defaultsFile=liquibase_dev.properties updateSQL
```



```sql
-- *********************************************************************
-- Update Database Script
-- *********************************************************************
-- Change Log: liquibase/update.xml
-- Ran at: 02.04.21 12:50
-- Against: HR@jdbc:oracle:thin:@127.0.0.1:1521/XEPDB1
-- Liquibase version: 4.3.0
-- *********************************************************************

-- Lock Database
UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 1, LOCKEDBY = 'DESKTOP-H61VFP7 (10.195.20.231)', LOCKGRANTED = TO_TIMESTAMP('2021-04-02

-- Changeset triggers/TRG_SECURE_EMPLOYEES.sql::raw::includeAll
CREATE OR REPLACE EDITIONABLE TRIGGER "HR"."TRG_SECURE_EMPLOYEES"
  BEFORE INSERT OR UPDATE OR DELETE ON employees
BEGIN
  p_secure_dml;
END TRG_SECURE_EMPLOYEES;

/
ALTER TRIGGER "HR"."TRG_SECURE_EMPLOYEES" DISABLE;

INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXT

-- Changeset triggers/TRG_UPDATE_JOB_HISTORY.sql::raw::includeAll
CREATE OR REPLACE EDITIONABLE TRIGGER "HR"."TRG_UPDATE_JOB_HISTORY"
  AFTER UPDATE OF job_id, department_id ON employees
  FOR EACH ROW
BEGIN
  p_add_job_history(:old.employee_id, :old.hire_date, sysdate,
                    :old.job_id, :old.department_id);
END;

/
ALTER TRIGGER "HR"."TRG_UPDATE_JOB_HISTORY" ENABLE;

INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXT

-- Changeset procedures/P_ADD_JOB_HISTORY.sql::raw::includeAll
CREATE OR REPLACE EDITIONABLE PROCEDURE "HR"."P_ADD_JOB_HISTORY"
  (  p_emp_id          job_history.employee_id%type --some comment
   , p_start_date      job_history.start_date%type
   , p_end_date        job_history.end_date%type
   , p_job_id          job_history.job_id%type
   , p_department_id   job_history.department_id%type
   )
IS
BEGIN
```

```
##      https://hub.liquibase.com                 ##
##                                                ##
####################################################
Starting Liquibase at 12:50:38 (version 4.3.0 #16 built at 2021-02-09 15:47+0000)
Liquibase command 'updatesql' was executed successfully.
PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR>
```

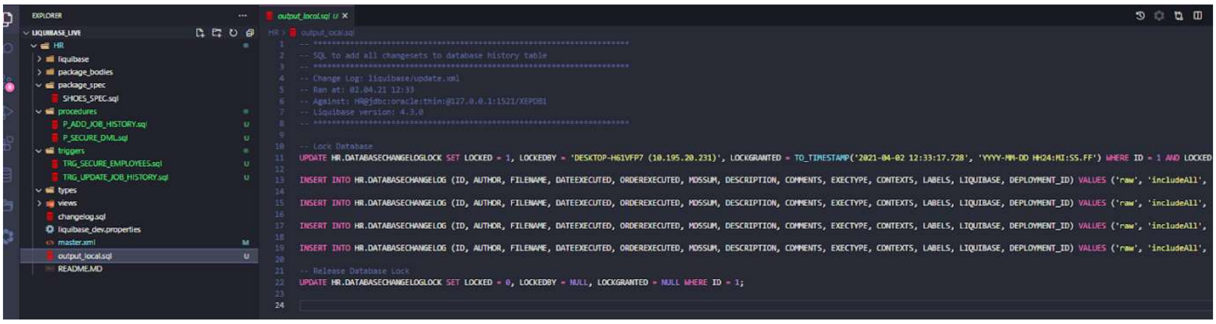
OK, bro. But this is not what we wanted! We already have these procedures and triggers in our database. We don’t want to create these objects again.

That’s where **ChangelogSync** and **ChangelogSyncSQL** commands come in!

Let’s run **ChangelogSyncSQL** to see what’s gonna happen:

```
liquibase —defaultsFile=liquibase_dev.properties ChangelogSyncSQL
```

The output SQL file is:



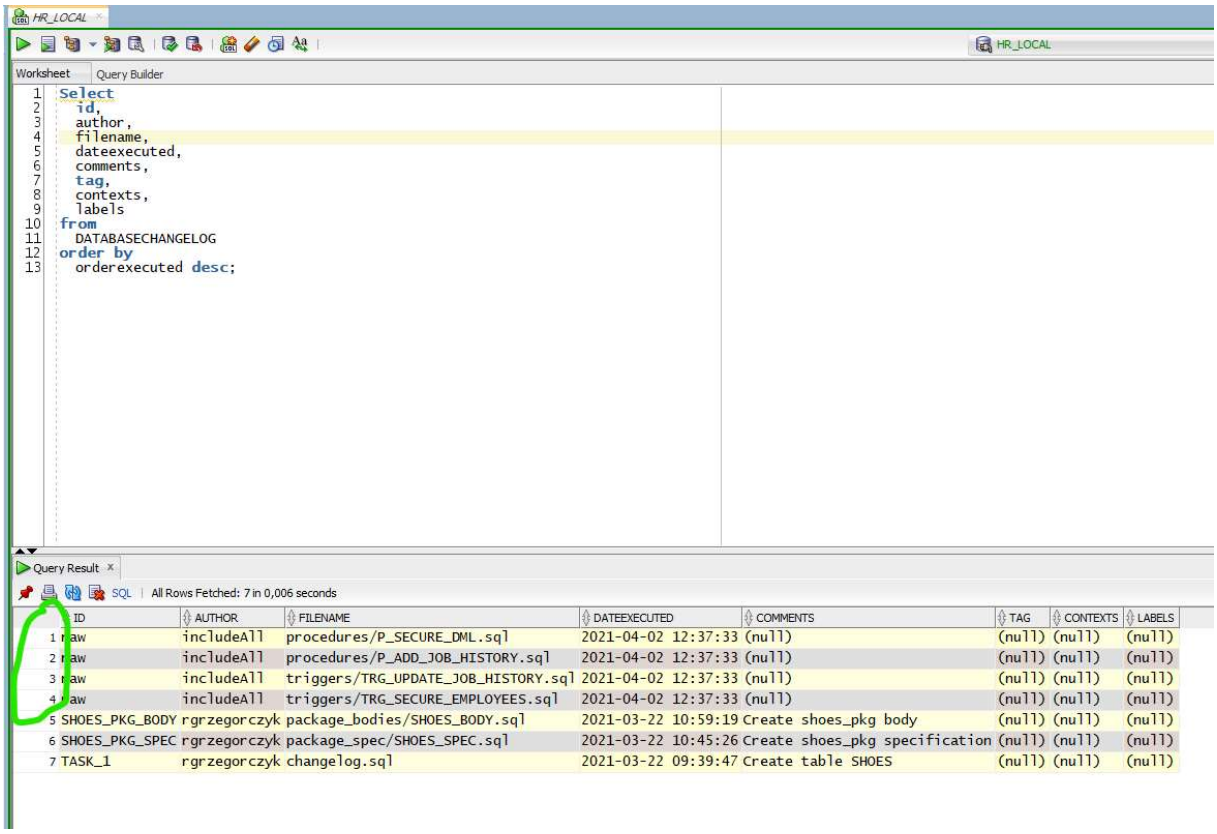
```sql
-- *********************************************************************
-- SQL to add all changesets to database history table
-- *********************************************************************
-- Change Log: liquibase/update.xml
-- Ran at: 02.04.21 12:33
-- Against: HR@jdbc:oracle:thin:@127.0.0.1:1521/XEPDB1
-- Liquibase version: 4.3.0
-- *********************************************************************

-- Lock Database
UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 1, LOCKEDBY = 'DESKTOP-H61VFP7 (10.195.20.231)', LOCKGRANTED = TO_TIMESTAMP('2021-04-02 12:33:17.728', 'YYYY-MM-DD HH24:MI:SS.FF') WHERE ID = 1 AND LOCKED

INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('raw', 'includeAll',

INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('raw', 'includeAll',

INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('raw', 'includeAll',

INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('raw', 'includeAll',

-- Release Database Lock
UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 0, LOCKEDBY = NULL, LOCKGRANTED = NULL WHERE ID = 1;

```

This is exactly what we wanted – just an SQL file with inserts in a **DATABASECHANGELOG** table. It will "tell" Liquibase that those objects were already created in the past, and there's no need to run them again.

Now, let's insert it to our Oracle database:

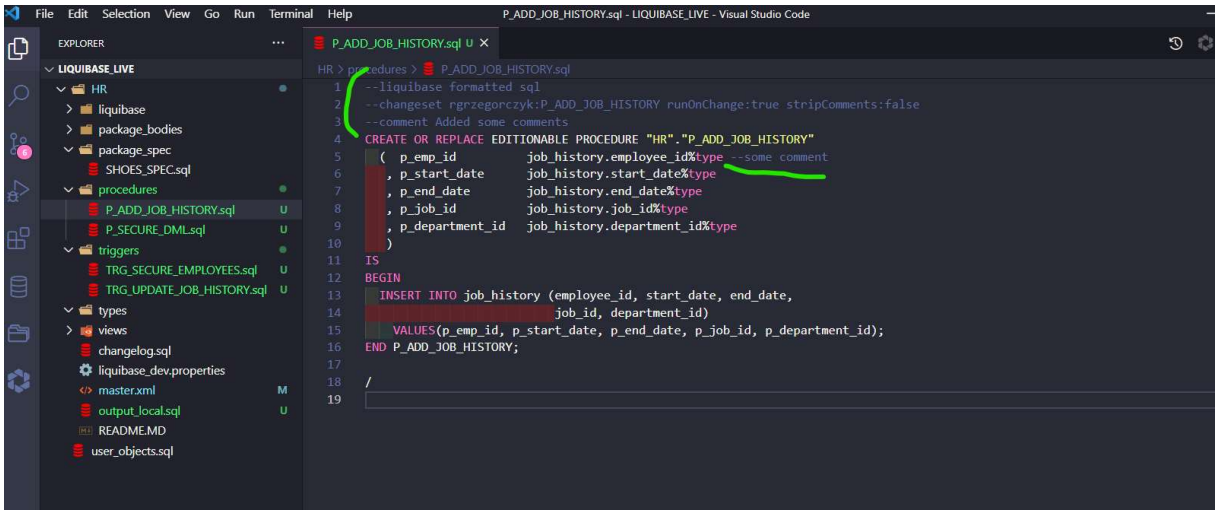
```
liquibase –defaultsFile=liquibase_dev.properties ChangelogSync
```

And we have 4 new changesets in the **DATABASECHANGELOG** table:

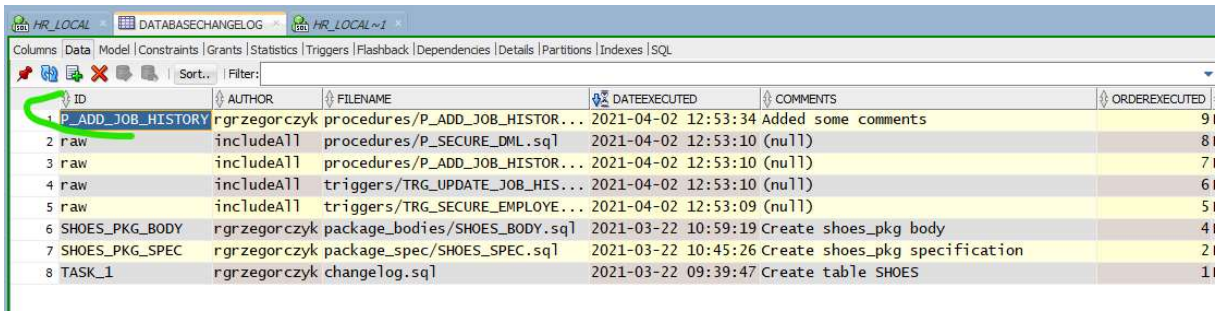


But what are these strange "raw" IDs? And why is the author called "includeAll"? Because this is the easiest and fastest way to move your existing project to Liquibase! And these changesets were created automatically.

If you'd like to do some changes, e.g. in **P_ADD_JOB_HISTORY**, just add a changeset – as you'd normally do when creating a new database object:



Then run the **Liquibase Update** command:



	ID	AUTHOR	FILENAME	DATEEXECUTED	COMMENTS	ORDEREXECUTED
1	P_ADD_JOB_HISTORY	rgrzegorczyk	procedures/P_ADD_JOB_HISTOR...	2021-04-02 12:53:34	Added some comments	9
2	raw	includeAll	procedures/P_SECURE_DML.sql	2021-04-02 12:53:10	(null)	8
3	raw	includeAll	procedures/P_ADD_JOB_HISTOR...	2021-04-02 12:53:10	(null)	7
4	raw	includeAll	triggers/TRG_UPDATE_JOB_HIS...	2021-04-02 12:53:10	(null)	6
5	raw	includeAll	triggers/TRG_SECURE_EMPLOYE...	2021-04-02 12:53:09	(null)	5
6	SHOES_PKG_BODY	rgrzegorczyk	package_bodies/SHOES_BODY.sql	2021-03-22 10:59:19	Create shoes_pkg body	4
7	SHOES_PKG_SPEC	rgrzegorczyk	package_spec/SHOES_SPEC.sql	2021-03-22 10:45:26	Create shoes_pkg specification	2
8	TASK_1	rgrzegorczyk	changelog.sql	2021-03-22 09:39:47	Create table SHOES	1

Changeset looks better now, right? With a proper author, ID, etc.

In the examples above, I showed you the easy way to add existing objects (which could be created or replaced) without creating changesets manually. In my opinion, it's the best way to install Liquibase if you have hundreds of objects in your existing database.

When it comes to objects which cannot be replaced, such as tables, we need to use a way described in the second scenario.

How to install Liquibase if you don't have lots of objects in your existing project

This option requires you to create changesets for objects and changes that were already executed to your database.

Objects which you create or replace

Add objects and remember to have paths to folders in your **master.xml** file – just like described in the first scenario.

Run **ChangelogSync** and have Liquibase automatically create changesets raw / includeAll / filename.

4	raw	includeAll	triggers/TRG_UPDATE_JOB_HIS...	2021-04-02 12:53:10	(null)
5	raw	includeAll	triggers/TRG_SECURE_EMPLOYE...	2021-04-02 12:53:09	(null)

Or, better way, create a changeset for every file like this:

P_SECURE_DML.sql U ×

HR > procedures > P_SECURE_DML.sql

```sql
--liquibase formatted sql
--changeset rgrzegorczyk:P_SECURE_DML runOnChange:true stripComments:false
--comment Created changeset for P_SECURE_DML
CREATE OR REPLACE EDITIONABLE PROCEDURE "HR"."P_SECURE_DML"
IS
BEGIN
  IF TO_CHAR (SYSDATE, 'HH24:MI') NOT BETWEEN '08:00' AND '18:00'
        OR TO_CHAR (SYSDATE, 'DY') IN ('SAT', 'SUN') THEN
  RAISE_APPLICATION_ERROR (-20205,
    'You may only make changes during normal office hours');
  END IF;
END P_SECURE_DML;

/
```

That's more work, sure, but you get better info in your logs:

HR_LOCAL | DATABASECHANGELOG | HR_LOCAL~1

Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

Sort.. | Filter: | Actions...

	ID	AUTHOR	FILENAME	DATEEXECUTED	COMMENTS	ORDEREXECUTED	EXECTYPE
1	P_SECURE_DML	rgrzegorczyk	procedures/P_SECURE_DML.sql	2021-04-02 13:12:49	Created changeset for P_SECURE_DML	10	EXECUTE
2	P_ADD_JOB_HISTORY	rgrzegorczyk	procedures/P_ADD_JOB_HISTOR...	2021-04-02 12:53:34	Added some comments	9	EXECUTE
3	raw	includeAll	procedures/P_SECURE_DML.sql	2021-04-02 12:53:10	(null)	8	EXECUTE
4	raw	includeAll	procedures/P_ADD_JOB_HISTOR...	2021-04-02 12:53:10	(null)	7	EXECUTE
5	raw	includeAll	triggers/TRG_UPDATE_JOB_HIS...	2021-04-02 12:53:10	(null)	6	EXECUTE
6	raw	includeAll	triggers/TRG_SECURE_EMPLOYE...	2021-04-02 12:53:09	(null)	5	EXECUTE
7	SHOES_PKG_BODY	rgrzegorczyk	package_bodies/SHOES_BODY.sql	2021-03-22 10:59:19	Create shoes_pkg body	4	RERAN
8	SHOES_PKG_SPEC	rgrzegorczyk	package_spec/SHOES_SPEC.sql	2021-03-22 10:45:26	Create shoes_pkg specification	2	EXECUTE
9	TASK_1	rgrzegorczyk	changelog.sql	2021-03-22 09:39:47	Create table SHOES	1	EXECUTE

Objects that cannot be created and/or replaced

What can you do with other types of objects like tables, indexes, etc.?

Once again, there are two ways:

1. Don't do anything with these objects but remember to always create changesets for every change in them, and add it to your **changelog.sql** file

(alter table, drop column, etc.) – I described how to do it in a previous part of this tutorial.

2. Create changesets and mark them as executed in the past.

Let's have a closer look at the second way.

I have a few tables that were created before implementing Liquibase:

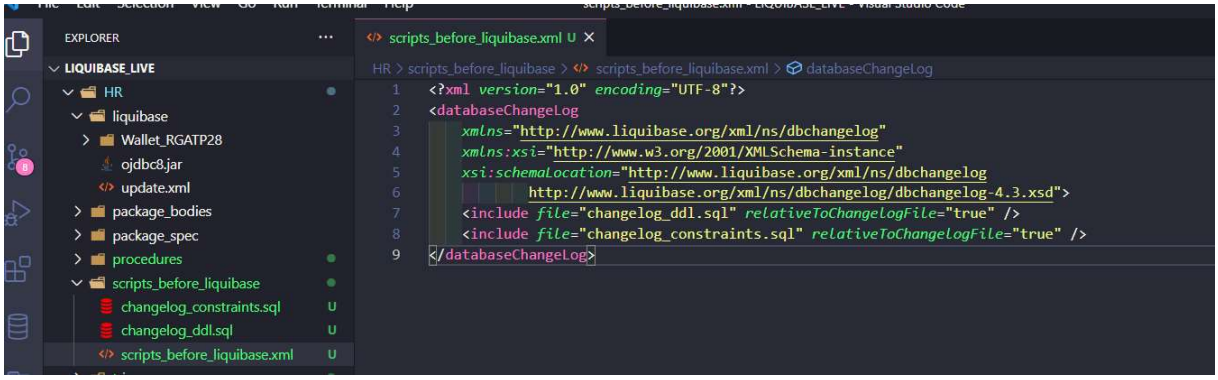
```
EMPLOYEES
JOBS
```

I create two changelog files in a new folder **HR/scripts_before_liquibase**.

```
changelog_ddl.sql
changelog_constraints.sql
```

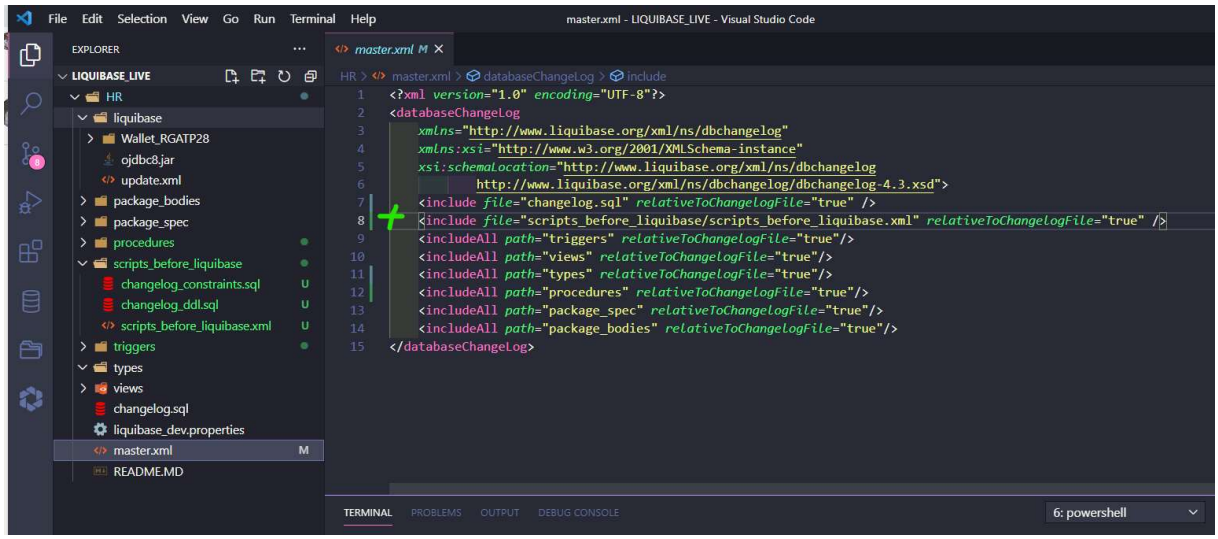
Also, I create an additional **scripts_before_liquibase.xml** file which will point to our changelogs.

The priority of "include file" is very important, as it tells Liquibase in which order to run scripts – first create tables, then, create constraints and indexes.

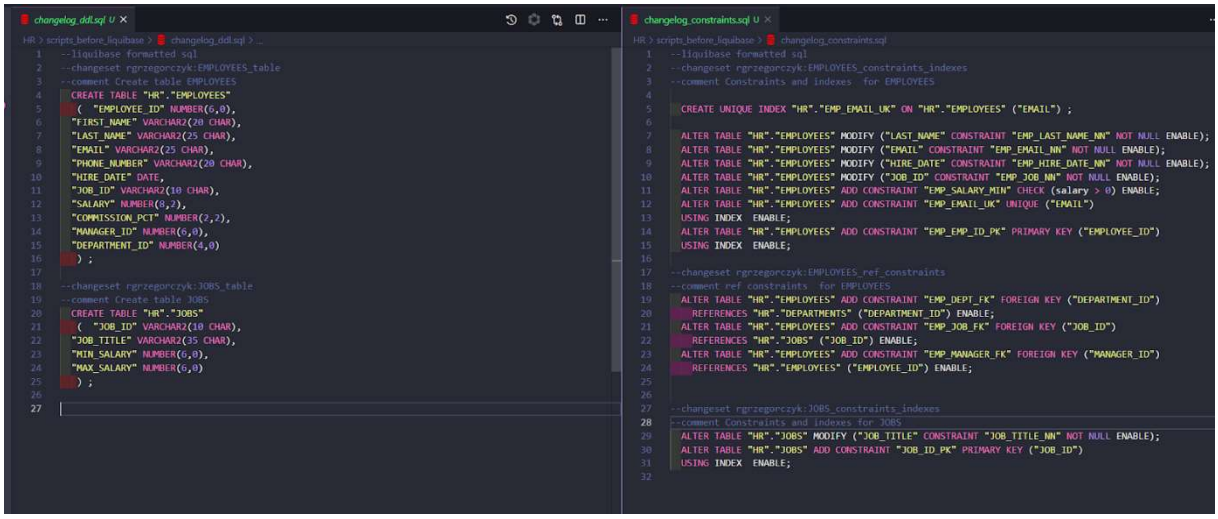


It's a good practice to have two files: one for creating tables, second for constraints. It will help you avoid conflicts when trying to create ref_constraint in a table which is gonna be created a few seconds later.

Remember to add a path to the **master.xml** file to your newly created XML file (HR/script_before_liquibase/scripts_before_liquibase.xml).



Now, create changesets for tables, constraints, etc.



OK, after we added all of our changesets, we will mark them as executed in the past.

Let's run **ChangelogSyncSQL** to preview, and then **ChangelogSync** to run SQL against database.

```
output_localsql U ×
HR > output_localsql > ...
1  -- *******************************************************************
2  -- SQL to add all changesets to database history table
3  -- *******************************************************************
4  -- Change Log: liquibase/update.xml
5  -- Ran at: 02.04.21 13:38
6  -- Against: HR@jdbc:oracle:thin:@127.0.0.1:1521/XEPDB1
7  -- Liquibase version: 4.3.0
8  -- *******************************************************************
9
10 -- Lock Database
11 UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 1, LOCKEDBY = 'DESKTOP-H61VFP7 (10.195.20.231)', LOCKGRANTED = TO_TIMESTAMP('2021-04-02 13:38:47.872', 'YYYY-MM-DD HH24:MI:SS.FF') WHERE ID = 1 AND LOCKED = 0;
12
13 INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('EMPLOYEES_table', 'rgrzegorczyk'
14
15 INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('JOBS_table', 'rgrzegorczyk', 'sc
16
17 INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('EMPLOYEES_constraints_indexes',
18
19 INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('EMPLOYEES_ref_constraints', 'rgr
20
21 INSERT INTO HR.DATABASECHANGELOG (ID, AUTHOR, FILENAME, DATEEXECUTED, ORDEREXECUTED, MD5SUM, DESCRIPTION, COMMENTS, EXECTYPE, CONTEXTS, LABELS, LIQUIBASE, DEPLOYMENT_ID) VALUES ('JOBS_constraints_indexes', 'rgrz
22
23 -- Release Database Lock
24 UPDATE HR.DATABASECHANGELOGLOCK SET LOCKED = 0, LOCKEDBY = NULL, LOCKGRANTED = NULL WHERE ID = 1;
25
26
```

```
TERMINAL   PROBLEMS   OUTPUT   DEBUG CONSOLE                                   6: powershell
##  Free schema change activity reports at       ##
##      https://hub.liquibase.com                ##
##                                               ##
###################################################
Starting Liquibase at 13:38:46 (version 4.3.0 #16 built at 2021-02-09 15:47+0000)
Liquibase command 'ChangelogSyncSQL' was executed successfully.
PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR> liquibase --defaultsFile=liquibase_dev.properties ChangelogSync
Liquibase Community 4.3.0 by Datical
###################################################
##                                               ##
##  |'|   (_)         (_)|                       ##
##  | |    _ _ _ _   _ |_   _ _ _ _   _ _ _ _    ##
##  | |   | |/ _ ` | | | | ` _ \ / _ / _|/ _ \   ##
##  | |__| | (_| | |_| | | | | | | (_| \__\ \_/ ##
##  \____/|\_, |\__,_|_|_|_/ \__,_|__/\___|      ##
##           |__|                                ##
##                                               ##
##  Get documentation at docs.liquibase.com      ##
##  Get certified courses at learn.liquibase.com ##
##  Free schema change activity reports at       ##
##      https://hub.liquibase.com                ##
##                                               ##
###################################################
Starting Liquibase at 13:40:15 (version 4.3.0 #16 built at 2021-02-09 15:47+0000)
Liquibase command 'ChangelogSync' was executed successfully.
PS C:\Users\pretius\OneDrive\MY_DEV\LIQUIBASE_LIVE\HR> []
```

And voila! All done!

Query Result

SQL | All Rows Fetched: 14 in 0,013 seconds

	ID	AUTHOR	FILENAME	DATEEXECUTED	COMMENTS	TAG	CONTEXTS	LABELS
1	JOBS_constraints_indexes	rgrzegorczyk	scripts_before_liquibase/changelog_constraints.sql	2021-04-02 13:40:17	Constraints and indexes for JOBS	(null)	(null)	(null)
2	EMPLOYEES_ref_constraints	rgrzegorczyk	scripts_before_liquibase/changelog_constraints.sql	2021-04-02 13:40:17	ref constraints for EMPLOYEES	(null)	(null)	(null)
3	EMPLOYEES_constraints_indexes	rgrzegorczyk	scripts_before_liquibase/changelog_constraints.sql	2021-04-02 13:40:17	Constraints and indexes for EMPLOYEES	(null)	(null)	(null)
4	JOBS_table	rgrzegorczyk	scripts_before_liquibase/changelog_ddl.sql	2021-04-02 13:40:17	Create table JOBS	(null)	(null)	(null)
5	EMPLOYEES_table	rgrzegorczyk	scripts_before_liquibase/changelog_ddl.sql	2021-04-02 13:40:17	Create table EMPLOYEES	(null)	(null)	(null)
6	P_SECURE_DML	rgrzegorczyk	procedures/P_SECURE_DML.sql	2021-04-02 13:12:49	Created changeset for P_SECURE_DML	(null)	(null)	(null)
7	P_ADD_JOB_HISTORY	rgrzegorczyk	procedures/P_ADD_JOB_HISTORY.sql	2021-04-02 12:53:34	Added some comments	(null)	(null)	(null)
8	raw	includeAll	procedures/P_SECURE_DML.sql	2021-04-02 12:53:10	(null)	(null)	(null)	(null)
9	raw	includeAll	procedures/P_ADD_JOB_HISTORY.sql	2021-04-02 12:53:10	(null)	(null)	(null)	(null)
10	raw	includeAll	triggers/TRG_UPDATE_JOB_HISTORY.sql	2021-04-02 12:53:10	(null)	(null)	(null)	(null)
11	raw	includeAll	triggers/TRG_SECURE_EMPLOYEES.sql	2021-04-02 12:53:09	(null)	(null)	(null)	(null)
12	SHOES_PKG_BODY	rgrzegorczyk	package_bodies/SHOES_BODY.sql	2021-03-22 10:59:19	Create shoes_pkg body	(null)	(null)	(null)
13	SHOES_PKG_SPEC	rgrzegorczyk	package_spec/SHOES_SPEC.sql	2021-03-22 10:45:26	Create shoes_pkg specification	(null)	(null)	(null)
14	TASK_1	rgrzegorczyk	changelog.sql	2021-03-22 09:39:47	Create table SHOES	(null)	(null)	(null)

Now, choose the way you prefer and implement database automation using Liquibase right now.

Liquibase tutorial: Summary

As you can see, by using Liquibase you can track everything during your database change release process.

However, all developers should stick to this workflow:

- Always add your changesets to a changelog (don't change anything without Liquibase!) – changeset should be unique combining AUTHOR:ID(task) and filename (file with your changelog)

- Verify the SQL you will execute (always run **updateSQL** before update command).
- Run database **update** command.
- Verify that the changeset or changesets were executed (check your DB objects and DATABASECHANGELOG table)

Here we end the basic Liquibase tutorial. However, stay tuned for the next articles! Here's what you can expect:

- Liquibase and GIT: How can many developers work effectively and conflict-free?
- Case study: How to use Liquibase without production access?
- How to create automated rollbacks using Liquibase?
- How to compare multiple database schemas using Liquibase?

https://pretius.com/blog/liquibase-tutorial/

https://www.java4coding.com/contents/liquibase/liquibase-tutorial

https://www.cloudbees.com/blog/liquibase-tutorial-manage-database-schema

https://www.baeldung.com/liquibase-refactor-schema-of-java-app

https://docs.liquibase.com/tools-integrations/springboot/using-springboot-with-maven.html

https://javadeveloperzone.com/spring-boot/spring-boot-liquibase-example/

https://www.techgeeknext.com/spring-boot/spring-boot-liquibase-example

https://codezup.com/integrate-liquibase-with-spring-boot-application/

https://reflectoring.io/database-migration-spring-boot-liquibase/

https://medium.com/techwasti/spring-boot-liquibase-to-manage-database-versioning-ffa2b713d29c

https://samerabdelkafi.wordpress.com/2020/10/26/liquibase-in-spring-boot-project/

https://javapointers.com/spring/spring-boot/add-liquibase-to-spring-boot-example/

https://roytuts.com/evolving-database-using-spring-boot-liquibase/

https://b-gyula.github.io/liquibase-doc/documentation/tutorials/oracle

https://oracle-base.com/articles/misc/liquibase-automating-your-database-deployments

https://www.broadleafcommerce.com/docs/core/5.0/appendix/managing-db-versions-migrations-with-liquibase

https://www.exoplatform.com/blog/safely-manage-the-database-for-your-exo-add-ons-with-liquibase-maven-plugin/

Crafting Interpreters

Game Programming Patterns

Computing for Biologists: Python Programming and Principles

Java Professional Interview Guide: Learn About Java Interview Questions and Practise Answering About Concurrency, JDBC, Exception Handling, Spring, and Hibernate

Problem Solving Using Data Structures and Algorithms: Develop your thought process to solve coding challenges; by Andrews Afful (Author)

Requirements Modeling and Coding: An Object-Oriented Approach ; Liping Liu (Author)

Learn Object Oriented Programming Using Java: An UML based

Murach's Oracle SQL and PL/SQL for Developers

Business Process driven Database Design: with Oracle PL/SQL

https://www.oracletutorial.com/plsql-tutorial/ --- ok

https://www.tutorialspoint.com/plsql/index.htm --- ya

https://www.plsqltutorial.com/