Build

Portability

This database is written in Java and therefore works on many platforms.

Environment

To run this database, a Java Runtime Environment (JRE) version 8 or higher is required. It it also possible to compile a standalone executable with experimental native image build.

To create the database executables, the following software stack was used. To use this database, it is not required to install this software however.

- Mac OS X and Windows
- Oracle JDK Version 8
- Eclipse
- Eclipse Plugins: Eclipse Checkstyle Plug-in, EclEmma Java Code Coverage
- Mozilla Firefox
- OpenOffice
- NSIS (Nullsoft Scriptable Install System)
- Maven

Building the Software

You need to install a JDK, for example the Oracle JDK version 8. Ensure that Java binary directory is included in the `PATH` environment variable, and that the environment variable `JAVA_HOME` points to your Java installation. On the command line, go to the directory `h2` and execute the following command:

```
build -?
```

For Linux and OS X, use `./build.sh` instead of `build`.

You will get a list of targets. If you want to build the `jar` file, execute (Windows):

```
build jar
```

To run the build tool in shell mode, use the command line option `-`:

```
./build.sh -
```

Using Apache Lucene

Apache Lucene 8.5.2 is used for testing.

Using Maven 2

Using a Central Repository

You can include the database in your Maven 2 project as a dependency. Example:

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <version>2.2.224</version>
</dependency>
```

New versions of this database are first uploaded to http://hsql.sourceforge.net/m2-repo/ and then automatically synchronized with the main Maven repository; however after a new release it may take a few hours before they are available there.

Maven Plugin to Start and Stop the TCP Server

A Maven plugin to start and stop the H2 TCP server is available from Laird Nelson at GitHub. To start the H2 server, use:

```
mvn com.edugility.h2-maven-plugin:1.0-SNAPSHOT:spawn
```

To stop the H2 server, use:

```
mvn com.edugility.h2-maven-plugin:1.0-SNAPSHOT:stop
```

Using Snapshot Version

To build a `h2-*-SNAPSHOT.jar` file and upload it the to the local Maven 2 repository, execute the following command:

```
build mavenInstallLocal
```

Afterwards, you can include the database in your Maven 2 project as a dependency:

```
<dependency>
  <groupId>com.h2database</groupId>
  <artifactId>h2</artifactId>
  <version>1.0-SNAPSHOT</version>
</dependency>
```

Native Image

There is an experimental support for compilation of native executables with native-image tool. To build an executable with H2 install GraalVM and use its updater to get the native-image tool:

```
gu install native-image
```

This tool can be used for compilation of native executables:

```
native-image --no-fallback -jar h2-VERSION.jar h2
```

Known limitations:

- If `--no-fallback` parameter was specified, system tray icon may not appear even if `-Djava.awt.headless=false` parameter of native-image tool was used, because native-image doesn't add all necessary configuration for working GUI.
- If `--no-fallback` parameter was specified, user-defined functions and triggers need an additional configuration.
- JAVA_OBJECT data type wasn't tested and may not work at all.
- Third-party loggers, ICU4J collators, and fulltext search weren't tested.

Using Eclipse

To create an Eclipse project for H2, use the following steps:

- Install Git and Eclipse.
- Get the H2 source code from Github:
  `git clone https://github.com/h2database/h2database`
- Download all dependencies:
  `build.bat download` (Windows)
  `./build.sh download` (otherwise)
- In Eclipse, create a new Java project from existing source code: `File, New, Project, Java Project, Create project from existing source`.
- Select the `h2` folder, click `Next` and `Finish`.
- To resolve `com.sun.javadoc` import statements, you may need to manually add the file `<java.home>/../lib/tools.jar` to the build path.

Translating

The translation of this software is split into the following parts:

- H2 Console: `src/main/org/h2/server/web/res/_text_*.prop`
- Error messages: `src/main/org/h2/res/_messages_*.prop`

To translate the H2 Console, start it and select Preferences / Translate. After you are done, send the translated `*.prop` file to the Google Group. The web site is currently translated using Google.

Submitting Source Code Changes

If you'd like to contribute bug fixes or new features, please consider the following guidelines to simplify merging them:

- Only use Java 8 features (do not use Java 9/10/etc) (see Environment).
- Follow the coding style used in the project, and use Checkstyle (see above) to verify. For example, do not use tabs (use spaces instead). The checkstyle configuration is in `src/installer/checkstyle.xml`.
- A template of the Eclipse settings are in `src/installer/eclipse.settings/*`. If you want to use them, you need to copy them to the `.settings` directory. The formatting options (`eclipseCodeStyle`) are also included.
- Please provide test cases and integrate them into the test suite. For Java level tests, see `src/test/org/h2/test/TestAll.java`. For SQL level tests, see SQL files in `src/test/org/h2/test/scripts`.
- The test cases should cover at least 90% of the changed and new code; use a code coverage tool to verify that (see above). or use the build target `coverage`.
- Verify that you did not break other features: run the test cases by executing `build test`.
- Provide end user documentation if required (`src/docsrc/html/*`).
- Document grammar changes in `src/main/org/h2/res/help.csv`
- Provide a change log entry (`src/docsrc/html/changelog.html`).
- Verify the spelling using `build spellcheck`. If required add the new words to `src/tools/org/h2/build/doc/dictionary.txt`.
- Run `src/installer/buildRelease` to find and fix formatting errors.
- Verify the formatting using `build docs` and `build javadoc`.
- Submit changes using GitHub's "pull requests". You'll require a free GitHub account. If you are not familiar with pull requests, please read GitHub's Using pull requests page.

For legal reasons, patches need to be public in the form of an issue report or attachment or in the form of an email to the group. Significant contributions need to include the following statement:

"I wrote the code, it's mine, and I'm contributing it to H2 for distribution multiple-licensed under the MPL 2.0, and the EPL 1.0 (https://h2database.com/html/license.html)."

Reporting Problems or Requests

Please consider the following checklist if you have a question, want to report a problem, or if you have a feature request:

- For bug reports, please provide a short, self contained, correct (compilable), example of the problem.
- Feature requests are always welcome, even if the feature is already on the issue tracker you can comment it. If you urgently need a feature, consider providing a patch.
- Before posting problems, check the FAQ and do a Google search.
- When got an unexpected exception, please try the Error Analyzer tool. If this doesn't help, please report the problem, including the complete error message and stack trace, and the root cause stack trace(s).
- When sending source code, please use a public web clipboard such as Pastebin or Mystic Paste to avoid formatting problems. Please keep test cases as simple and short as possible, but so that the problem can still be reproduced. As a template, use: HelloWorld.java. Method that simply call other methods should be avoided, as well as unnecessary exception handling. Please use the JDBC API and no external tools or libraries. The test should include all required initialization code, and should be started with the main method.
- For large attachments, use a public storage such as Google Drive.
- Google Group versus issue tracking: Use the Google Group for questions or if you are not sure it's a bug. If you are sure it's a bug, you can create an issue, but you don't need to (sending an email to the group is enough). Please note that only few people monitor the issue tracking system.
- For out-of-memory problems, please analyze the problem yourself first, for example using the command line option `-XX:+HeapDumpOnOutOfMemoryError` (to create a heap dump file on out of memory) and a memory analysis tool such as the Eclipse Memory Analyzer (MAT).
- It may take a few days to get an answers. Please do not double post.

Automated Build

This build process is automated and runs regularly. The build process includes running the tests and code coverage, using the command line `./build.sh jar testCI`. The results are available on CI workflow page.

Generating Railroad Diagrams

The railroad diagrams of the SQL grammar are HTML, formatted as nested tables. The diagrams are generated as follows:

- The BNF parser (`org.h2.bnf.Bnf`) reads and parses the BNF from the file `help.csv`.
- The page parser (`org.h2.server.web.PageParser`) reads the template HTML file and fills in the diagrams.
- The rail images (one straight, four junctions, two turns) are generated using a simple Java application.

To generate railroad diagrams for other grammars, see the package `org.h2.jcr`. This package is used to generate the SQL-2 railroad diagrams for the JCR 2.0 specification.