Frequently Asked Questions

I Have a Problem or Feature Request

Please read the support checklist.

Are there Known Bugs? When is the Next Release?

Usually, bugs get fixes as they are found. There is a release every few weeks. Here is the list of known and confirmed issues:

- When opening a database file in a timezone that has different daylight saving rules: the time part of dates where the daylight saving doesn't match will differ. This is not a problem within regions that use the same rules (such as, within USA, or within Europe), even if the timezone itself is different. As a workaround, export the database to a SQL script using the old timezone, and create a new database in the new timezone.
- Old versions of Tomcat and Glassfish 3 set most static fields (final or non-final) to `null` when unloading a web application. This can cause a `NullPointerException`. In Tomcat >= 6.0 this behavior can be disabled by setting the system property `org.apache.catalina.loader.WebappClassLoader.ENABLE_CLEAR_REFERENCES=false`. A known workaround is to put the `h2*.jar` file in a shared `lib` directory (`common/lib`). Tomcat 8.5 and newer versions don't clear fields and don't have such property.
- Some problems have been found with right outer join. Internally, it is converted to left outer join, which does not always produce the same results as other databases when used in combination with other joins. This problem is fixed in H2 version 1.3.

For a complete list, see Open Issues.

Is this Database Engine Open Source?

Yes. It is free to use and distribute, and the source code is included. See also under license.

Is Commercial Support Available?

No, currently commercial support is not available.

How to Create a New Database?

By default, a new database is automatically created if it does not yet exist when embedded URL is used. See Creating New Databases.

How to Connect to a Database?

The database driver is `org.h2.Driver`, and the database URL starts with `jdbc:h2:`. To connect to a database using JDBC, use the following code:

```
Connection conn = DriverManager.getConnection("jdbc:h2:~/test", "sa", "");
```

Where are the Database Files Stored?

When using database URLs like `jdbc:h2:~/test`, the database is stored in the user directory. For Windows, this is usually `C:\Documents and Settings\<userName>` or `C:\Users\<userName>`. If the base directory is not set (as in `jdbc:h2:./test`), the database files are stored in the directory where the application is started (the current working directory). When using the H2 Console application from the start menu, this is `<Installation Directory>/bin`. The base directory can be set in the database URL. A fixed or relative path can be used. When using the URL `jdbc:h2:file:./data/sample`, the database is stored in the directory `data` (relative to the current working directory). The directory is created automatically if it does not yet exist. It is also possible to use the fully qualified directory name (and for Windows, drive name). Example: `jdbc:h2:file:C:/data/test`

What is the Size Limit (Maximum Size) of a Database?

See Limits and Limitations.

Is it Reliable?

That is not easy to say. It is still a quite new product. A lot of tests have been written, and the code coverage of these tests is higher than 80% for each package. Randomized stress tests are run regularly. But there are probably still bugs that have not yet been found (as with most software). Some features are known to be dangerous, they are only supported for situations where performance is more important than reliability. Those dangerous features are:

- Disabling the transaction log or FileDescriptor.sync() using LOG=0 or LOG=1.
- Using the transaction isolation level `READ_UNCOMMITTED` (`LOCK_MODE 0`) while at the same time using multiple connections.
- Disabling database file protection using (setting `FILE_LOCK` to `NO` in the database URL).
- Disabling referential integrity using `SET REFERENTIAL_INTEGRITY FALSE`.

In addition to that, running out of memory should be avoided. In older versions, OutOfMemory errors while using the database could corrupt a databases.

This database is well tested using automated test cases. The tests run every night and run for more than one hour. But not all areas of this database are equally well tested. When using one of the following features for production, please ensure your use case is well tested (if possible with automated test cases). The areas that are not well tested are:

- Platforms other than Windows, Linux, Mac OS X, or runtime environments other than Oracle / OpenJDK 7, 8, 9.
- The features `AUTO_SERVER` and `AUTO_RECONNECT`.
- Cluster mode, 2-phase commit, savepoints.
- Fulltext search.
- Operations on LOBs over 2 GB.
- The optimizer may not always select the best plan.
- Using the ICU4J collator.

Areas considered experimental are:

- The PostgreSQL server
- Clustering (there are cases were transaction isolation can be broken due to timing issues, for example one session overtaking another session).
- Compatibility modes for other databases (only some features are implemented).
- The soft reference cache (`CACHE_TYPE=SOFT_LRU`). It might not improve performance, and out of memory issues have been reported.

Some users have reported that after a power failure, the database cannot be opened sometimes. In this case, use a backup of the database or the Recover tool. Please report such problems. The plan is that the database automatically recovers in all situations.

Why is Opening my Database Slow?

To find out what the problem is, use the H2 Console and click on "Test Connection" instead of "Login". After the "Login Successful" appears, click on it (it's a link). This will list the top stack traces. Then either analyze this yourself, or post those stack traces in the Google Group.

Other possible reasons are: the database is very big (many GB), or contains linked tables that are slow to open.

My Query is Slow

Slow `SELECT` (or `DELETE, UPDATE, MERGE`) statement can have multiple reasons. Follow this checklist:

- Run `ANALYZE` (see documentation for details).
- Run the query with `EXPLAIN` and check if indexes are used (see documentation for details).
- If required, create additional indexes and try again using `ANALYZE` and `EXPLAIN`.
- If it doesn't help please report the problem.

H2 is Very Slow

By default, H2 closes the database when the last connection is closed. If your application closes the only connection after each operation, the database is opened and closed a lot, which is quite slow. There are multiple ways to solve this problem, see Database Performance Tuning.

Column Names are Incorrect?

For the query `SELECT ID AS X FROM TEST` the method `ResultSetMetaData.getColumnName()` returns `ID`, I expect it to return `X`. What's wrong?

This is not a bug. According the JDBC specification, the method `ResultSetMetaData.getColumnName()` should return the name of the column and not the alias name. If you need the alias name, use `ResultSetMetaData.getColumnLabel()`. Some other database don't work like this yet (they don't follow the JDBC specification). If you need compatibility with those databases, use the Compatibility Mode.

This also applies to DatabaseMetaData calls that return a result set. The columns in the JDBC API are column labels, not column names.

Float is Double?

For a table defined as `CREATE TABLE TEST(X FLOAT)` the method `ResultSet.getObject()` returns a `java.lang.Double`, I expect it to return a `java.lang.Float`. What's wrong?

This is not a bug. According the JDBC specification, the JDBC data type `FLOAT` is equivalent to `DOUBLE`, and both are mapped to `java.lang.Double`. See also Mapping SQL and Java Types - 8.3.10 FLOAT.

Use REAL or FLOAT(24) data type for `java.lang.Float` values.

How to Translate this Project?

For more information, see Build/Translating.

How to Contribute to this Project?

There are various way to help develop an open source project like H2. The first step could be to translate the error messages and the GUI to your native language. Then, you could provide patches. Please start with small patches. That could be adding a test case to improve the code coverage (the target code coverage for this project is 90%, higher is better). You will have to develop, build and run the tests. Once you are familiar with the code, you could implement missing features from the feature request list. I suggest to start with very small features that are easy to implement. Keep in mind to provide test cases as well.