How To / FAQ

How to install and run HSQLDB

- Download the hsqldb.zip file from the download page to your local computer.
- Expand the .zip using any program that can handle zip files. You can also do this with jar -xf hsqldb.zip
- You need a Java Runtime Environment (JRE) or Java Development Kit (JDK) to run HSQLDB.
- Browse the documentation included in the zip package. Start with 'index.html'.

How to compile HSQLDB

You don't need to. A compiled JAR of HSQLDB is included in the zip in the /lib folder. This runs under Java 11 and later and support Java module system. A second Jar, hsqldb-jdk8.jar works with JRE 8 and later. If you want to re-compile HSQLDB, you will need a JDK. You can use JDK 8, 11, 17 or 21. See the Guide and the documentation in the /build directory.

How to create a new database

A new database is created automatically if it does not yet exist. Just connect to the not-yet-existing database using the jdbc:hsqldb:file:«database-path» URL (should replace the last part with the path you want) with the user 'sa' (or any other name) and a password (can be an empty string). You will use this name and password to connect again.

How to start programming JDBC / HSQLDB

HSQLDB comes with PDF and HTML documentation, with example program source code that can help programmers who are new to JDBC programming.

Basic sample programs are in the /src/org/hsqldb/sample folder.

Source code of test programs are useful examples of how to use different features of JDBC and SQL. Check the sources in the /src/org/hsqldb/test folder.

SQL test scripts are in the /testrun folder and offer extensive examples of SQL statements.

Where to get more Documentation

HSQLDB has a standard JDBC interface. HSQLDB specific JDBC documentation is included in the /doc/apidocs folder.

There are also many books available on JDBC programming.

HSQLDB is covered in hundreds of books on Java programming. Search Google Books for "HSQLDB"

How to use HSQLDB inside Eclipse, etc.

To use HSQLDB at design-time in Eclipse, NetBeans or other tools, you usually require the plug-in for databases that comes with the development environment. You usually need to add a reference to the HSQLDB jar to the environment. Also you normally need to register the JDBC driver (which is part of the hsqldb.jar) with the environment. A dedicated plugin is already available for Eclipse.

How to upgrade from an old version to the current version

It is recommended that you close the databases with the SHUTDOWN command before they are opened with version 2.x. If the database is not read-only, it will be upgraded to the latest version. Databases made with versions earlier than 2.0 must be upgraded before opening with version 2.7. This is described in the Guide.

If you use CACHED tables, the procedures in the System Management and Deployment Issues section of the Guide should be followed. A SHUTDOWN COMPACT after the upgade is recommended.

Note that an upgrade is a one-way process, so please always keep a backup of the old database.

License

- May I use HSQLDB in a commercial product?

  Yes. HSQLDB is Open Source and free to use in any commercial product so long as the terms of the Licenses are met. The Licenses of HSQLDB and Hypersonic SQL (on which a few parts of HSQLDB are based) are both based on the new BSD License.

Reliability, Performance and Deployment

- Does HSQLDB store all data in memory. Doesn't memory run out as a result?

  It stores all data in memory only if you want to. By default, CREATE TABLE results in a memory table, as this is the best type for smaller tables. For larger tables, use CREATE CACHED TABLE and adjust the cache size to suite your memory use requirements (as little as 8MB or so). See the System Management and Deployment Issues chapter of the Guide. There is no simple rule and no imposition on the part of HSQLDB as maximum flexibility is allowed using only a couple of settings. A popular use of HSQLDB is for OLAP, ETL, and data mining applications where huge Java memory allocations are used to hold millions of rows of data in memory.

- How does HSQLDB compare to other RDBMS engines in SQL support / JDBC support

  HSQLDB has very extensive support for SQL-2023. This support nearly matches the Advanced level of the old SQL-92 Standard and the Core level of the new Standard, plus over 150 optional features. SQL support is more extensive than all open-source database products and includes features that are not yet supported in most closed-source, commercial products. JDBC support is comprehensive and extends to all the features that are supported by the core SQL capabilities of the engine.

- How does HSQLDB compare to other RDBMS engines in speed

  This is something that you can measure. Overall, with disk tables, HSQLDB is faster, or at least comparable in speed to the fastest non-java or java open-source RDBMS engines. In addition, HSQLDB supports fast, persistent, memory tables in a database which are much faster than traditional disk tables.
  You can use the performance tests supplied with HSQLDB. One test, JDBCBench.java is a standard TPC-B implementation that measures speed and reliability of multi-threaded access. HSQLDB is extremely fast in the MVCC mode in this test. Another test, TestCacheSize.java, is a single-threaded test for speed of INSERT, UPDATE, DELETE and SELECT operations with hundreds of thousands of rows. Some comparisons have been posted by users in our mailing lists and on the web.

  Any specific SELECT query speed issues can normally be resolved with slight modifications to the query or by adding appropriate indexes. See the Guide for details.

- How solid is HSQLDB

  HSQLDB 2.x comes after version 2.0 and uses the experience gained from extensive SQL compatibility tests and stress tests by application vendors that use HSQLDB in their products. Persistence in 2.x is an improved and hardened version of the persistence engine of 1.8 which was in use for over 5 years.

- How solid is HSQLDB when a machine crashes

  HSQLDB employs a redo log for data recovery. All the changes to the database are reflected in this log. Extensive user tests have demonstrated this mechanism to be effective and fail-safe in most cases. For added security, you can backup the database files while the engine is running using the BACKUP DATABASE command.

  By default, a FileDescriptor.sync() call is made every 500 milliseconds on the redo log file. If the machine or the Java process is likely to crash often, you can reduce this down to 20 milliseconds for more frequent sync() calls. You can also specify 0 to force a sync() on each commit. This setting can be changed with "SET FILES WRITE DELAY MILLIS m" or the equivalent connection property.

  With all the rest of the database files, calls to sync() are made at all critical points to ensure the files are consistent both after a shutdown and after a crash.

- What about multithreading?

  HSQLDB 2.x is fully multithreaded. Both the core engine and the Server. In the MVCC mode, multiple threads can write to the same table while other threads perform SELECT queries.

- What are the limitations of the database (size of columns, number of tables, rows...)?

  There is no imposed limitation. Number of columns, tables, indexes, size of columns and so on is limited only by the memory. For example, a user reported using a SELECT statement with 41 LEFT OUTER JOIN clauses on a huge database for a data mining application.

- My database runs out of memory / How much memory does a database need?

  If only memory tables (CREATE TABLE or CREATE MEMORY TABLE) are used then the database is limited by the memory. A minimum of about 100 bytes plus the actual data size are required for each row. If you use CREATE CACHED TABLE, then the size of the table is not limited by the memory beyond a certain minimum size. The data and indexes of cached tables are saved to disk. With text tables, indexes are memory resident but the data is cached to disk.

- What is the biggest known HSQLDB database?

  The current default size limit of an HSQLDB database is 8TB for all CACHED tables combined (this can be extended to 256TB with a connection property). The size limit for each TEXT table is 256GB. In addition, maximum total lob size is 64TB. If you use large MEMORY tables, memory is only limited by the allocated JVM memory, which can be several GB on modern machines and 64bit operating systems. We have performed extensive tests with the latest versions using the TestCacheSize and other test programs inserting millions of rows and resulting in data files of up to 16 GB and larger LOB sizes. Users have reported the use of databases with up to 900 million rows.

- After the program is finished, there are a lot statements in the *.log file. Why?

  The database was not shut down properly. When you restart the database, the *.log file will be processed and an automatic checkpoint will be performed. No data committed before the last sync() (see under machine crash above) will be lost. To avoid this, use the SQL command "SHUTDOWN" when your application has finished with the database.

  The statements that make up the database are saved in the *.script file (mostly CREATE statements and INSERT statements for memory tables). Only the data of cached tables (CREATE CACHED TABLE) is stored in the *.data file. Also all data manipulation operations are stored in the *.log file (mostly DELETE/INSERT) for crash recovery. When the SHUTDOWN or CHECKPOINT command is issued to a database, then the *.script file is re-created and becomes up-to-date. The .log file is deleted. When the database is restarted, all statements of the *.script file are executed first and new statements are appended to the .log file as the database is used. In addition, there are *.bakup and *.lobs files.

- The size of the .data file grows in operation although the total row count for CACHED tables has not gone up. Why?

  From version 2.6.0 you can enable SET FILES SPACE TRUE and empty spaces are tracked much better than old versions. HSQLDB tracks and reuses empty spaces left after DELETE or during UPDATE operations. If the .data file is still too fragmented, both CHECKPOINT DEFRAG and SHUTDOWN COMPACT commands will remove the empty spaces.

- I need transaction isolation levels beyond read-committed

  HSQLDB 2.x supports READ COMMITTED and SERIALIZABLE isolation levels. It supports both lock-based and multiversion (MVCC) transaction models, and is fully multithreaded in all modes.

  See the Guide for a fuller discussion of all the issues.