Architecture

Introduction

H2 implements an embedded and standalone ANSI-SQL89 compliant SQL engine on top of a B-tree based disk store.

As of October 2013, Thomas is still working on our next-generation storage engine called MVStore. This will in time replace the B-tree based storage engine.

Top-down Overview

Working from the top down, the layers look like this:

- JDBC driver.
- Connection/session management.
- SQL Parser.
- Command execution and planning.
- Table/Index/Constraints.
- Undo log, redo log, and transactions layer.
- B-tree engine and page-based storage allocation.
- Filesystem abstraction.

JDBC Driver

The JDBC driver implementation lives in `org.h2.jdbc, org.h2.jdbcx`

Connection/session management

The primary classes of interest are:

Package	Description
org.h2.engine.Database	the root/global class
org.h2.engine.SessionInterface	abstracts over the differences between embedded and remote sessions
org.h2.engine.Session	local/embedded session
org.h2.engine.SessionRemote	remote session

Parser

The parser lives in `org.h2.command.Parser`. It uses a straightforward recursive-descent design.

See Wikipedia Recursive descent parser page.

Command execution and planning

Unlike other databases, we do not have an intermediate step where we generate some kind of IR (intermediate representation) of the query. The parser class directly generates a command execution object. Then we run some optimisation steps over the command to possibly generate a more efficient command.

The primary packages of interest are:

Package	Description
org.h2.command.ddl	Commands that modify schema data structures
org.h2.command.dml	Commands that modify data

Table/Index/Constraints

One thing to note here is that indexes are simply stored as special kinds of tables.

The primary packages of interest are:

Package	Description
org.h2.table	Implementations of different kinds of tables
org.h2.index	Implementations of different kinds of indices

Undo log, redo log, and transactions layer

We have a transaction log, which is shared among all sessions. See also
https://en.wikipedia.org/wiki/Transaction_log https://h2database.com/html/grammar.html#set_log

We also have an undo log, which is per session, to undo an operation (an update that fails for example) and to rollback a transaction. Theoretically, the transaction log could be used, but for simplicity, H2 currently uses it's own "list of operations" (usually in-memory).

With the MVStore, this is no longer needed (just the transaction log).

B-tree engine and page-based storage allocation.

The primary package of interest is `org.h2.store`.

This implements a storage mechanism which allocates pages of storage (typically 2k in size) and also implements a b-tree over those pages to allow fast retrieval and update.

Filesystem abstraction.

The primary class of interest is `org.h2.store.FileStore`.

This implements an abstraction of a random-access file. This allows the higher layers to treat in-memory vs. on-disk vs. zip-file databases the same.