H2database 源码分析

陶封邑 2020K8009937014

功能分析与建模

1.什么是 H2database？

H2database 是一个开源的、纯 Java 编写的嵌入式数据库引擎。它包含了 JDBC 的 API，拥有 ODBC 的驱动程序，并且提供了一个 Web 控制平台来对数据库进行操作。得益于其纯 Java 编写的代码组成，H2database 不受平台的限制，可以运行在 Windows、Linux 等不同的操作系统上。

H2database 可以实现内存模式与硬盘模式两种数据库运行模式。内存模式中数据仅在内存中运行，每一个用例执行完随即还原到初始状态，这种方式适合用于测试。硬盘模式中数据被持久化的保存为单个文件，可以储存少量的结构化数据。

H2database 拥有嵌入式和服务器两种使用模式。嵌入式指将 H2database 嵌入进 Java 项目中，与项目应用同时运行，这种方式的 H2database 性能最高。服务器模式指 H2 server 将 H2database 启动，应用通过与 H2 server 连接进而访问、使用数据库。

H2database 最大的特点在于“轻量”上，它的所用功能与实现被包含在一个 2MB 左右的 jar 包中，可以十分轻易的嵌入需要数据库的项目中。“麻雀虽小，五脏俱全”，H2database 也包含各种功能，如：数据加密、全文搜索等等。

2.H2database 的主要模块与功能。

（1）API 包：用于用户定义扩展的接口，例如触发器和用户定义的聚合函数。

（2）BNF 包：BNF 范式的解析器和相关工具，包含为 BNF 自动填补上下文的类。

（3）COMMAND 包：SQL 语句的解析器和它的基础类，包含实现 DDL 语法、DML 语法的两个类以及实现查询的类。

（4）COMPRESS 包：用于数据无损压缩。

（5）CONSTAINT 包：用于数据库约束，如检查约束、惟一约束和引用约束。

（6）ENGINE 包：包含数据库的高级类和不适合放在另一个子包中的类，如用户以及用户相关的设置、远程链接的工具、管理员。

（7）EXPRESSION 包：数学式、简单值的表达式，包含实现聚集函数、数据分析操作和窗口函数、状态表达式、其他函数（包含表值函数）的类。

（8）FULLTEXT 包：实现原生全文搜索与 Lucene 全文搜索。

（9）INDEX 包：实现各种表索引，以及用于索引中导航的标签。

（10）JDBC 包：实现 JDBC API，包含实现 JDBC 数据库的元数据 API 的类。

（11）JDBCX 包：实现拓展的 JDBC API，如 JDBC 的连接池。

（12）JMX 包：实现 JAVA 的管理拓展。

（13）MESSAGE 包：实现跟踪(日志记录工具)功能、与错误消息相关的工具。

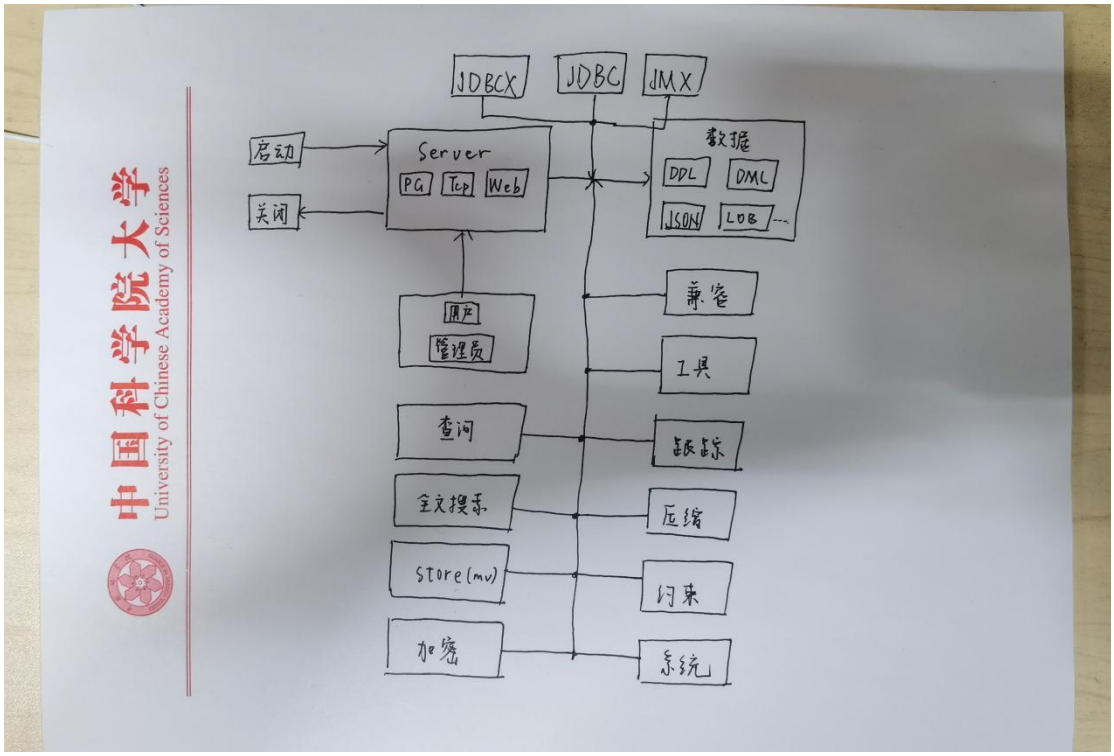
（14）MODE 包：实现与其他数据库(例如 MySQL)兼容的实用程序。

（15）MVSTORE 包：实现树形持久储存，包含用于 cache、h2database 中 mvstore 的相关助手、R-树、事务处理事件中 mvstore 的相关助手、数据类型和序列化/反序列化的类。

（16）RESULT 包：实现行与内部结果集，如远程结果集、抽象提取的结果集。

（17）SCHEMA 包：实现模式和存储在模式中的对象，如表示用户定义的函数或别名、序列和常量。
（18）SECURITY 包：实现数据安全与加密（例如加密安全哈希算法），包含身份验证相关的类。
（19）SERVER 包：实现 PostgreSQL、H2 Console、TCP 客户端。
（20）STORE 包：实现存储抽象、抽象文件系统，例如带有缓存的文件、将值转换为字节数组，包含用于将文件储存到硬盘（并通过 JAVA 语句访问）的系统、抽象的加密文件系统、将文件完全保存在内存（非 JAVA 堆）的系统（可选择压缩）、记录所有写操作并能重复调用这些操作的文件系统、当文件被关闭重新打开并重新尝试操作的文件系统、将文件分割成多个小文件的文件系统、基于 ZIP 的文件系统的类。
（21）TABLE 包：实现表和相关表元数据，例如 JBDC 相关的链接表、列解析器。
（22）TOOLS 包：实现各种工具，例如 CSV 工具、shell 工具。
（23）UTIL 包：实现内部实用工具，例如计算 MVTable 的关键词、值和页占用的内存量的工具，包含用于解析处理 GEOMETRY、JSON 数据的类。
（24）VALUE 包：实现数据类型和数据值，例如数据转换和比较的方法，包含用于 LOB 数据值的类。
（25）DRIVER 包：实现一个 JDBC 驱动。

3.H2database 的主要流程。

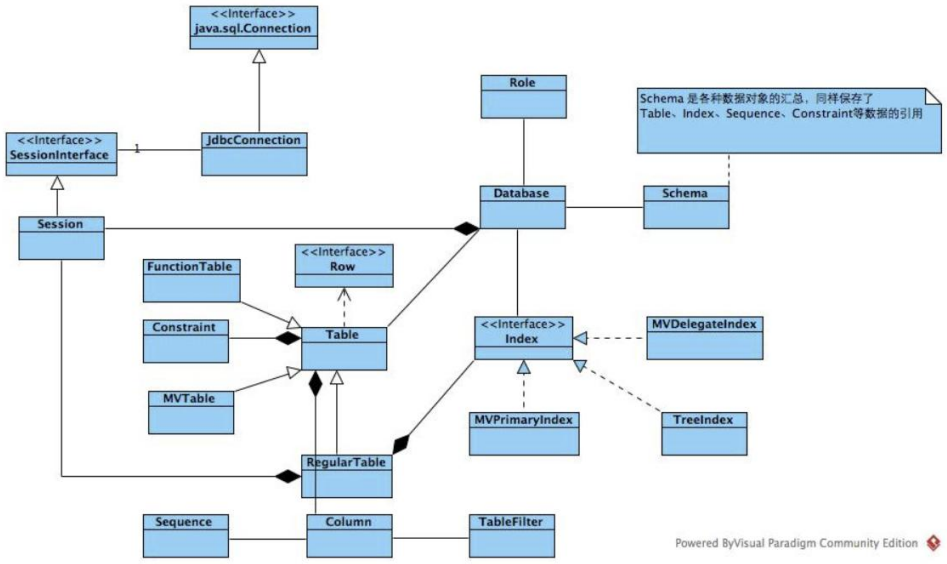


4.着重分析的模块。

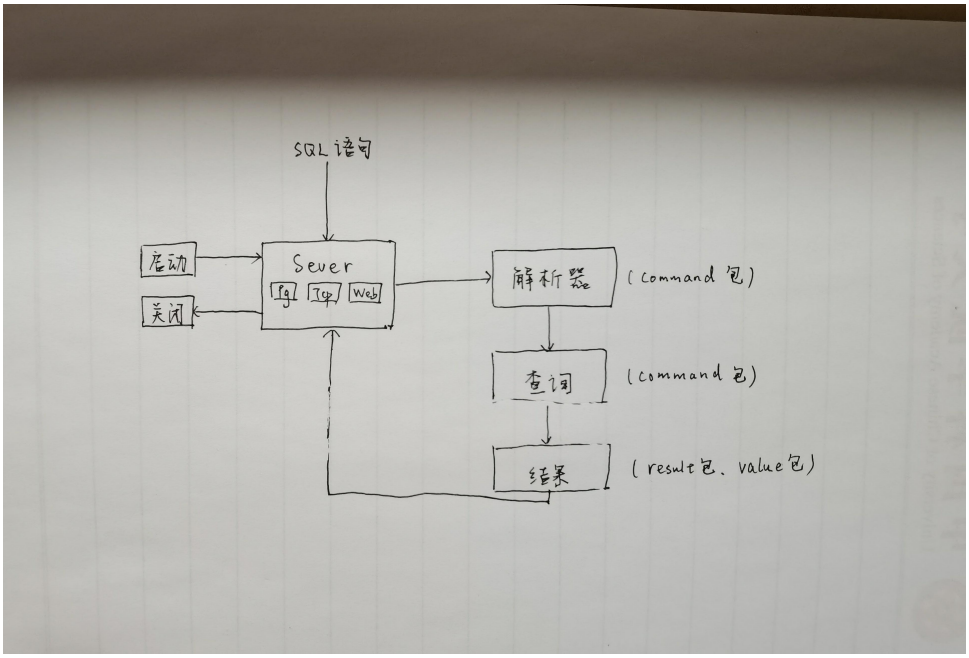
接下来将主要对查询功能进行分析。

核心流程设计分析

H2database 作为一个数据库，它应该包含组织、存储和管理数据的基本功能。它最主要的类的关系如下图所示：



这一节我们将从管理数据入手，分析 H2database 的核心功能“查询”的具体实现流程，查询的流程可以简化成下列关系：



在介绍查询之前首先介绍用户与数据库交流的工具：结构化查询语言 SQL（STRUCTURED QUERY LANGUAGE）。SQL 一种是用于访问和处理数据库的 ANSI 的标准的计算机语言，它在 1986 年 10 月由美国国家标准局（ANSI）通过的数据库语言美国标准，接着，国际标准化组织（ISO）颁布了 SQL 正式国际标准。可以把 SQL 分为两个部分：数据操作语言（DML）和数据定义语言（DDL）。

H2database 在接收到 SQL 语句之后，会先将该语句进行解析，再将解析之后的语句用

于对数据库的操作。SQL 语句由 org.h2.command.Parser 类解析，其中解析 DML 语言的相关方法在 org.h2.command.dml 包中；解析 DDL 语言的相关方法在 org.h2.command.ddl 包中。

“查询”操作就是对数据库操作中重要的一个，H2database 实现查询的 Command 中的 executeQuery 方法：

```
public ResultInterface executeQuery(long var1, boolean var3) {
```

在 executeQuery 中调用了 query：

```
try {
    ResultInterface var26 = this.query(var1);
    var7 = !var26.isLazy();
    ResultInterface var27;
    if (var6.getMode().charPadding == CharPadding.IN_RESULT_SETS) {
        var27 = ResultWithPaddedStrings.get(var26);
        return var27;
```

进一步查看 CommandContainer 和 Query 中的 query 方法，发现其中会出现大量的 queryWithoutCache 字样：

```
if (this.isUnion()) {
    return this.queryWithoutCacheLazyCheck(var1, var3);
} else {
```

最终可以在 Select 中找到 queryWithoutCache 方法：

```
protected ResultInterface queryWithoutCache(long var1, ResultTarget var3) {
```

由此我们得到一个具体的查询调用关系：Command.executeQuery→CommandContainer.query→Query.query→Select.queryWithoutCache。

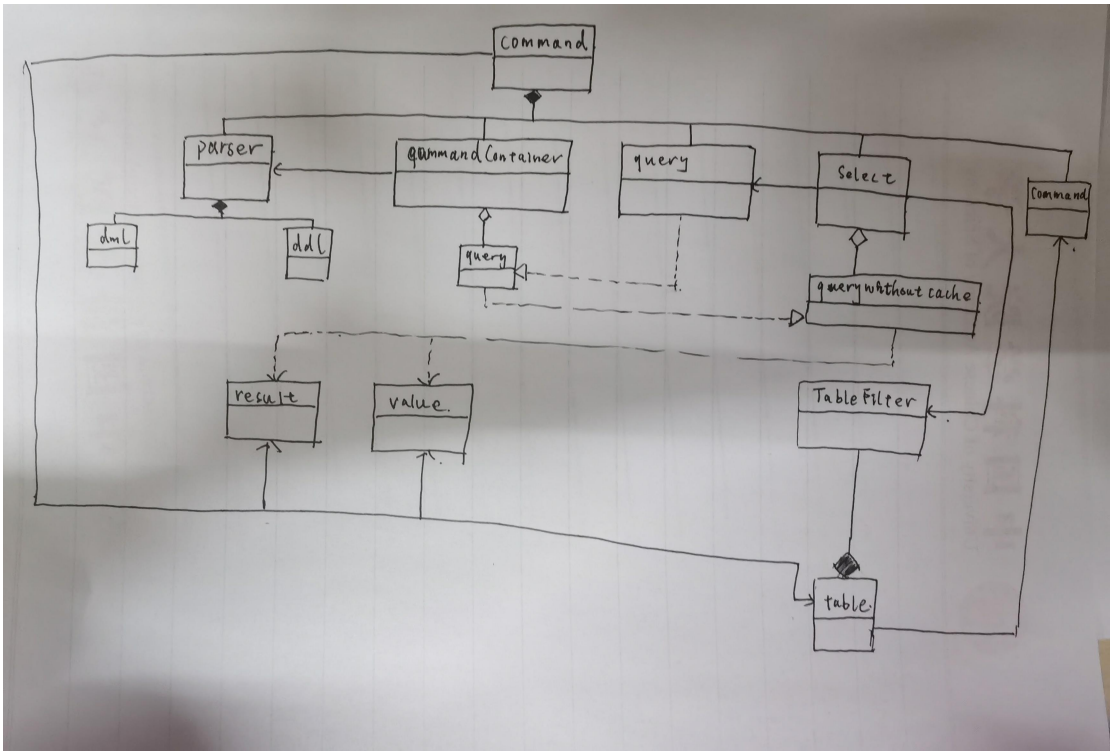
但因为能力所限，并没有在这个过程中找到具体比对关键字的方法，查阅资料得知，真正执行对比的方法为：

```
 * A string type.
 */
public class StringDataType implements DataType {

    public static final StringDataType INSTANCE = new StringDataType();

    @Override
    public int compare(Object a, Object b) {
        return a.toString().compareTo(b.toString());
    }
```

Select 包含一个 TableFilter（猜测也可能是一个过滤条件），负责查询数据库记录，然后 isConditionMet 负责判断条件是否符合，符合条件后，拼接成一个 row，然后返回。

总结一下一个用于查询的 SQL 语句在 H2database 中具体执行过程。首先数据库调用 parser 对 SQL 语句进行解析，发现它是一个查询语句，即该 select 语法会创建一个 Select 对象。此后就会调用 executeQuery 进行查找，这个查找过程中包含 TableFilter 用于过滤与记录，StringDataType 用于比对。

到此，H2database 的查询过程告一段落，查询相关的类关系如下表示：




```java
protected Value[] fetchNextRow() {
    while(Select.this.topTableFilter.next()) {
        Select.this.setCurrentRowNumber(this.rowNumber + 1L);
        if (this.forUpdate) {
            if (!Select.this.isConditionMetForUpdate()) {
                continue;
            }
        } else if (!Select.this.isConditionMet()) {
            continue;
        }

        ++this.rowNumber;
        Value[] var1 = new Value[this.columnCount];

        for(int var2 = 0; var2 < this.columnCount; ++var2) {
            Expression var3 = (Expression)Select.this.expressions.get(var2);
            var1[var2] = var3.getValue(Select.this.getSession());
        }

        return var1;
    }

    return null;
}
```

高级设计意图

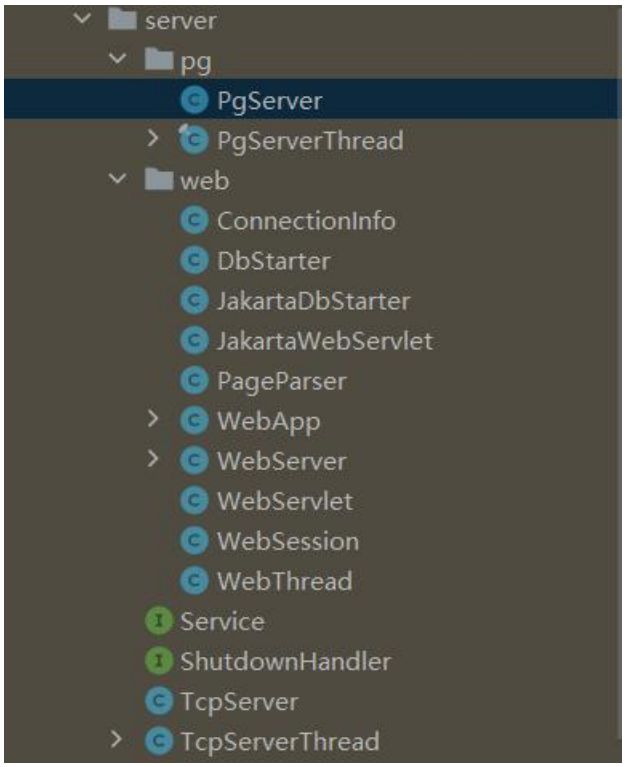
这里介绍H2database中体现的两种高级设计意图：工厂模式、策略模式，当然H2database在整个设计实现过程中还使用了其他设计意图。

工厂模式是一种创建设计模式，它在父类中提供了创建对象的接口，但允许子类更改将要创建的对象的类型。工厂方法模式建议将直接对象构造调用（使用 new 操作符）替换为对特殊工厂方法的调用。对象仍然是通过 new 操作符创建的，但它是从工厂方法内部调用的。

H2database 中最能体现工厂模式的是它对 Server 的实现。在 server 包里 H2database 首先实现了一个叫做 Service 的父类，定义了 init()、getURL()、start()、listen()、stop()、isRunning()、getAllowOthers()、getName()、getType()、getPort()、isDaemon()等方法，这些方法的具体实现则交由更具体的 Service 类型来完成。

```
package org.h2.server;

import java.sql.SQLException;

3 implementations
public interface Service {
```

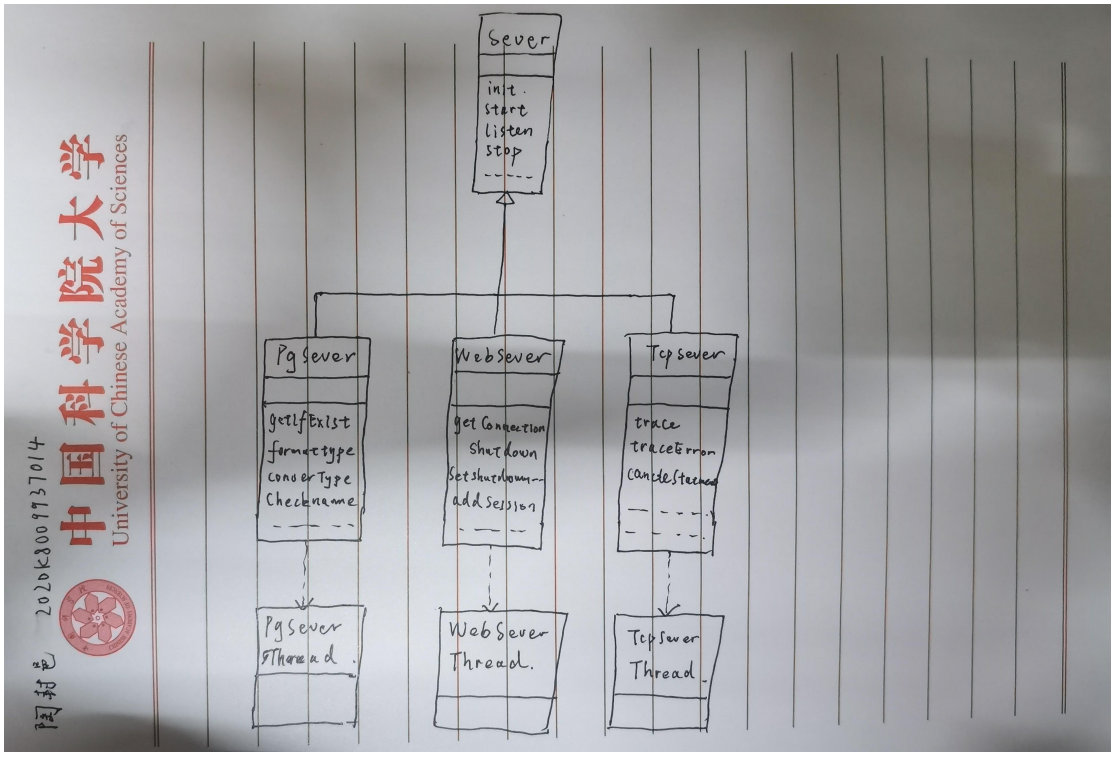
H2database 中实现了三种不同的 Server，分别为 PgServer、WebServer、TcpServer，其中 Pg、Web、Tcp 表示三种网络连接方式，通过实现三种具体子类，就可以启动上面三个 Server。



另外，启动 Server 的调用顺序是 init() → start() →listen()，用于启动的类 org.h2.tools.Server 类。

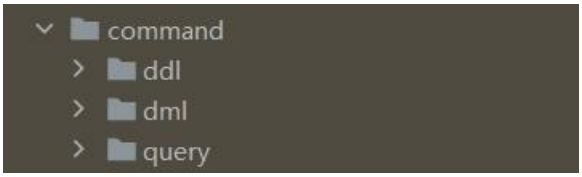
```java
public class Server extends Tool implements Runnable, ShutdownHandler {
    private final Service service;
    private Server web;
    private Server tcp;
    private Server pg;
    private ShutdownHandler shutdownHandler;
    private boolean started;

    public Server() { this.service = null; }
```

三个 sever 的关系可以如下图表示：



策略模式指针对一组算法，将每一个算法封装到具有共同接口的独立的类中，从而使得它们可以相互替换。策略模式使得算法可以在不影响到客户端的情况下发生变化。一个类定义了多种行为,并且这些行为在这个类的操作中以多个条件语句的形式出现。将相关的条件分支移入它们各自的类中以代替这些条件语句。

H2database 中体现策略模式的一点是它对 SQL 语句的解析过程。SQL 语句分为 DML 语句与 DDL 语句两种，两种语句的权限、语法等都不相同，因此 H2database 在实现 SQL 解析功能的时候将分析 DML 语句与 DDL 语句需要的类分别封装到两个包中。



比如 CREATE TABLE 语句对应 org.h2.command.ddl.CreateTable 类，INSERT 语句对应 org.h2.command.dml.Insert 类，SHOW 语句在 Parser 类中当成 SELECT 语句，对应 org.h2.command.dml.Select 类，相应的 CREATE TABLE 语句属于 DDL 语句，INSERT 语句与 SHOW 语句属于 DML 语句。

```java
public final class Insert extends CommandWithValues implements ResultTarget {
    private Table table;
    private Column[] columns;
    private Query query;
    private long rowNumber;
    private boolean insertFromSelect;
    private Boolean overridingSystem;
    private HashMap<Column, Expression> duplicateKeyAssignmentMap;
    private Value[] onDuplicateKeyRow;
    private boolean ignore;
    private ResultTarget deltaChangeCollector;
    private DataChangeDeltaTable.ResultOption deltaChangeCollectionMode;

    public Insert(SessionLocal var1) { super(var1); }

    public void setCommand(Command var1) {
        super.setCommand(var1);
        if (this.query != null) {
            this.query.setCommand(var1);
        }

    }
```

总结

总而言之 H2database 模型层次结构清晰，注释也很完善，是学习 java 的好工程。但因为 java 对我而言是一门全新的语言，根据一学期的学习也终究只是掌握了一些皮毛。同时数据库相关知识也是下学期才学习，这里提前查询资料难免有很多疑惑的地方。因此在阅读代码过程中遇见了不少问题，得出的结论也不尽人意，很多问题没有解答。但是总体过程是愉快的，不仅加深了对 java 和数据库的理解，也了解到了开源工程的有趣之处。