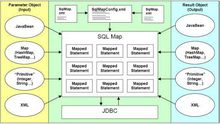
iBATIS一词来源于“[internet](https://baike.baidu.com/item/internet)”和“abatis”的组合，是一个由Clinton Begin在2001年发起的[开放源代码](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E4%BB%A3%E7%A0%81)项目。于2010年6月16号被谷歌托管，改名为[MyBatis](https://baike.baidu.com/item/MyBatis)。是一个基于[SQL](https://baike.baidu.com/item/SQL)映射支持Java和·NET的[持久层](https://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "_blank)框架。

一站式

iBATIS提供的[持久层](https://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "_blank)框架包括SQL Maps和Data Access Objec

[](https://baike.baidu.com/pic/ibatis/10787592/0/bf096b63f6246b602c18af9bebf81a4c500fa2ce?fr=lemma&ct=single)ibatis框架

ts（[DAO](https://baike.baidu.com/item/DAO)），同时还提供一个利用这个框架开发的JPetStore实例。

相对[Hibernate](https://baike.baidu.com/item/Hibernate)和[Apache](https://baike.baidu.com/item/Apache" \t "_blank)OJB等“一站式”ORM解决方案而言，ibatis 是一种“半自动化”的ORM实现。

iBATIS 目前提供了三种语言实现的版本，包括：Java、[.NET](https://baike.baidu.com/item/.NET" \t "_blank)以及[Ruby](https://baike.baidu.com/item/Ruby" \t "_blank)。右图是iBATIS的架构图：

目前主流

所谓“半自动化”，可能理解上有点生涩。纵观目前主流的 ORM(**对象关系映射**)，无论 [Hibernate](https://baike.baidu.com/item/Hibernate)还是[Apache](https://baike.baidu.com/item/Apache/6265" \t "_blank) OJB，都对数据库结构提供了较为完整的封装，提供了从[POJO](https://baike.baidu.com/item/POJO" \t "_blank)到数据库表的全套映射机制。程序员往往只需定义好了POJO 到数据库表的映射关系，即可通过 Hibernate或者OJB 提供的方法完成[持久层](https://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "_blank)操作。程序员甚至不需要对 SQL 的熟练掌握，Hibernate/OJB 会根据制定的存储逻辑，自动生成对应的 SQL 并调用 JDBC 接口加以执行。

大多数情况下（特别是对新项目，新系统的开发而言），这样的机制无往不利，大有一统天下的势头。但是，在一些特定的环境下，这种一站式的解决方案却未必灵光。

开发

在笔者的系统咨询工作过程中，常常遇到以下情况：

1． 系统的部分或全部数据来自现有数据库，出于安全考虑，只对开发团队提供几条Select SQL（或[存储过程](https://baike.baidu.com/item/%E5%AD%98%E5%82%A8%E8%BF%87%E7%A8%8B" \t "_blank)）以获取所需数据，具体的表结构不予公开。

2． 开发规范中要求，所有牵涉到业务逻辑部分的数据库操作，必须在数据库层由存储过程实现（就笔者工作所面向的金融行业而言，[工商银行](https://baike.baidu.com/item/%E5%B7%A5%E5%95%86%E9%93%B6%E8%A1%8C" \t "_blank)、[中国银行](https://baike.baidu.com/item/%E4%B8%AD%E5%9B%BD%E9%93%B6%E8%A1%8C)、[交通银行](https://baike.baidu.com/item/%E4%BA%A4%E9%80%9A%E9%93%B6%E8%A1%8C)，都在开发规范中严格指定）

3． 系统数据处理量巨大，性能要求极为苛刻，这往往意味着我们必须通过经过高度优化的SQL语句（或存储过程）才能达到系统性能设计指标。

面对这样的需求，再次举起 Hibernate 大刀，却发现刀锋不再锐利，甚至无法使用，奈何？恍惚之际，只好再摸出JDBC 准备拼死一搏……，说得未免有些凄凉，直接使用 JDBC进行数据库操作实际上也是不错的选择，只是拖沓的数据库访问代码，乏味的字段读取操作令人厌烦。

半自动化

[编辑](javascript:;)

“[半自动化](https://baike.baidu.com/item/%E5%8D%8A%E8%87%AA%E5%8A%A8%E5%8C%96)”的ibatis，却刚好解决了这个问题。这里的“半自动化”，是相对Hibernate等提供了全面的数据库封装机制的“全自动化”[ORM](https://baike.baidu.com/item/ORM) 实现而言，“全自动”ORM 实现了 [POJO](https://baike.baidu.com/item/POJO) 和数据库表之间的映射，以及 SQL 的自动生成和执行。而ibatis 的着力点，则在于POJO 与 SQL之间的映射关系。也就是说，ibatis并不会为程序员在运行期自动生成 SQL 执行。具体的 SQL 需要程序员编写，然后通过映射[配置文件](https://baike.baidu.com/item/%E9%85%8D%E7%BD%AE%E6%96%87%E4%BB%B6" \t "_blank)，将SQL所需的参数，以及返回的结果字段映射到指定 POJO。

通常在如下场景和条件下，选择ibatis, 将更有助于发挥ibatis在[持久层](https://baike.baidu.com/item/%E6%8C%81%E4%B9%85%E5%B1%82" \t "_blank)的优越性：

1. 知道怎样操作10种以上的数据库

2. 可配置的caching(包括从属)

3. 支持DataSource、local transaction management和global transaction

4. 简单的XML配置文档

5. 支持[Map](https://baike.baidu.com/item/Map/2592475" \t "_blank), [Collection](https://baike.baidu.com/item/Collection/80124), [List](https://baike.baidu.com/item/List)和简单类型包装(如Integer, String)

6. 支持[JavaBeans](https://baike.baidu.com/item/JavaBeans" \t "_blank)类(get/set 方法)

7. 支持复杂的对象映射(如populating lists, complex object models)

8.对象模型从不完美(不需要修改)

9. 数据模型从不完美(不需要修改)

10. 你已经知道SQL，为什么还要学习其他东西

全自动

[编辑](javascript:;)

使用ibatis 提供的ORM机制，对业务逻辑实现人员而言，面对的是纯粹的 Java对象，

这一层与通过 Hibernate 实现 ORM 而言基本一致，而对于具体的数据操作，Hibernate

会自动生成SQL 语句，而ibatis 则要求开发者编写具体的 SQL 语句。相对Hibernate等

“全自动”[ORM](https://baike.baidu.com/item/ORM)机制而言，ibatis 以 SQL开发的工作量大和数据库移植性上差为代价，为系统

设计提供了更大的自由空间。作为“全自动”ORM实现的一种有益补充，ibatis 的出现显

得别具意义。

发展

[编辑](javascript:;)

ibatis本是apache的一个开源项目，2010年这个项目由apache software foundation 迁移到了google code，并且改名为mybatis。