[](http://image.baidu.com/i?ct=503316480&z=0&tn=baiduimagedetail&word=%D6%D0%B9%FA%D2%C6%B6%AF+logo&in=8288&cl=2&cm=1&sc=0&lm=-1&pn=4&rn=1&di=508018540&ln=140)

**中国移动专利申请**

**技术交底书**

|  |  |
| --- | --- |
| **公司编号** |  |
| **发明名称** | 一种面向Oracle与Hive、Hbase的脚本语言转换方法和实现 |
| **申报单位** | 中国移动（深圳）有限公司 |
| **申报类型** | 发明 |
| **发明人** | 刘晟、陈立彬、成楼 |
| **技术联系人** | 刘晟 [liusheng@chinamobilesz.com](mailto:liusheng@chinamobilesz.com) 13923805275 |

|  |
| --- |
| **注意事项**  1．技术联系人应为深入了解本申请提案技术方案的技术人员，如交底书撰写人，负责向专利审核人员和代理人解释技术细节、修改交底书、审核申请文件等工作, 请务必填全技术联系人的姓名、E-mail、手机。  2．请按照集团公司提供的本技术交底书模板逐项填写，除交底书第八部分为可选项外，其他均为必须填写的内容。填写不全的专利申请提案，集团公司不予立案。  3．专利申请不要求已具体实现或实施，形成完整的技术方案即可提交申请，特别是需要向合作方公开、向标准提案或以其他方式公开的重要技术构思应在公开前尽早申请。  4．技术交底书文件命名要求：发明名称＋短横线（半角）＋交底书＋版本号，例：一种短消息群发方法-交底书v1.doc |

**中国移动通信集团公司**

一、发明名称

一种面向Oracle与Hive、Hbase的脚本语言转换方法和实现

**注：发明名称反映了本技术方案的核心内容和类型。**

二、技术领域

本发明涉及数据库领域，更具体地说，是一种面向Oracle与Hive、Hbase的脚本语言转换方法和实现。

**注：在列出的技术领域中选择了与本技术方案最为相关的领域。**

**。**

三、现有技术的技术方案

在传统的IT业界，大型应用系统多使用专业的关系型数据库产品构建其数据库系统，其中，Oracle数据库所占据的市场份额是最大的，Oracle是以高级结构化查询语言(SQL)为基础的大型关系数据库，通俗地讲它是用方便逻辑管理的语言操纵大量有规律数据的集合,是目前最流行的客户/服务器(Client/Server)体系结构的数据库之一。

在一定的数据规模下，Oracle为系统的数据存储、访问和管理提供了较好的支持，但在信息化日益加深的今天，系统所需要承载和处理的数据规模呈几何级数激增，目前已经进入了所谓的“大数据”时代，大型企业级系统面临“数据瓶颈”，面对这些问题，以Oracle为代表的大型关系数据库的局限性日益明显，其升级和维护成本大幅提高。

基于上述情况，云计算和大数据技术及产品应时而生，并被越来越多不同规模、不同行业的企业所接受，作为在海量数据环境下Oracle数据库的替代方案，其中，Hadoop项目集是目前最备受业界青睐的开源云计算和大数据解决方案，本发明涉及到的子项目除了Hadoop核心，还有Hbase和Hive。

Hadoop是由Apache基金会所开发的一个分布式系统基础架构，HDFS和MapReduce是Hadoop的两大核心。而整个Hadoop的体系结构主要是通过HDFS来实现对分布式存储的底层支持的，并且它会通过MapReduce来实现对分布式并行任务处理的程序支持。HDFS有高容错性的特点，并且设计用来部署在低廉的（low-cost）硬件上；而且它提供高传输率（high throughput）来访问应用程序的数据，适合那些有着超大数据集（large data set）的应用程序。HDFS放宽了（relax）POSIX的要求，可以以流的形式访问（streaming access）文件系统中的数据。MapReduce是一个简单易用的软件框架，基于它可以将任务分发到由上千台商用机器组成的集群上，并以一种高容错的方式并行处理大量的数据集，实现Hadoop的并行任务处理功能。

HBase是一个分布式的、面向列的开源数据库，该技术来源于 Fay Chang 所撰写的Google论文“Bigtable：一个结构化数据的分布式存储系统”。就像Bigtable利用了Google文件系统（File System）所提供的分布式数据存储一样，HBase在Hadoop之上提供了类似于Bigtable的能力。利用HBase技术可在廉价PC Server上搭建起大规模结构化存储集群

Hive是建立在 Hadoop 上的数据仓库基础构架。它提供了一系列的工具，可以用来进行数据提取转化加载（ETL），这是一种可以存储、查询和分析存储在 Hadoop 中的大规模数据的机制。Hive 定义了简单的类 SQL 查询语言，称为 HiveQL，它允许熟悉 SQL 的用户查询数据。同时，这个语言也允许熟悉 MapReduce 开发者的开发自定义的 mapper 和 reducer 来处理内建的 mapper 和 reducer 无法完成的复杂的分析工作。

在目前的IT业界，将大型关系数据库（如Oracle）和HBase、Hive等非传统数据系统混用已不鲜见，随着各行业大数据化程度的日益加深，这种解决方案将成为一种主流趋势。

**注：描述了现有技术的具体实现，尤其是说明了在现有技术中的实现中的使用，使人能够符合逻辑地得出第四部分中现有技术存在的缺点。**

四、现有技术的缺点及本申请提案要解决的技术问题

使用上述混合数据存储和处理解决方案，可以根据系统各部分不同的功能需求使用不同的数据处理方式，达到扬长避短的效果，但同时也带来了一些问题，因为架构和实现方式的不同，Oracle、HBase和Hive都有自己的脚本语言支持应用系统的数据访问和管理需求。 Oracle使用结构化查询语言SQL，HBase使用自定义的Shell进行建表、查询、更新数据等操作，而Hive则使用类SQL语言HiveQL实现数据操作，这种差异性带来的问题主要有：

1. 竖井式数据管理：应用程序需要针对Oracle、HBase和Hive编写不同的数据访问代码，增加了数据整合的困难度，另外，程序变更时也需要根据不同的底层数据访问方式进行修改，提高了维护的难度。
2. 历史遗留系统升级困难：很多企业在考虑将IT资产向大数据平台移植时遇到的一个大问题就是遗留系统的改造，而其核心就是数据访问层代码的修改。大型企业级IT系统的代码量常常是百万级的，这意味着对系统的任何变更都可能涉及到大量的修改，而数据访问层的升级涉及的主要是传统关系数据库（如Oracle）SQL的改写，如果完全由程序员手工改写将需要极大的工作量。

针对上述问题，本发明提出了一个可以实现Oracle数据库SQL语言向HBase Shell和HiveQL语言的转换方法，进而屏蔽差异性，支持无缝的数据整合和系统升级。

**注：逐一说明了各项现有技术存在的缺点，与第三部分现有技术的方案的描述相呼应；并对应于现有技术的缺点，提出本发明要解决的技术问题。**

五、本申请提案的技术方案的详细阐述

本专利技术方案主要是支持于将Oracle SQL转化为HBase Shell和HiveQL，下面将结合实例，对本专利技术方案作详细阐述。

和大多数关系型数据库一样，Oracle的数据访问和管理通常是通过执行一系列SQL（Structured Query Language，结构化查询语言）语句完成的。SQL语句按照其功能可分为三类：

1. DDL，数据定义语言，用于定义、更新、删除数据库对象，通常的数据库对象包含表、视图、索引、约束、存储过程、触发器、角色、用户等，本专利中主要指数据表。
2. DML，数据操纵语言，用于操作数据库中的数据。通常是对数据的查询、插入、更新、删除。
3. DCL，数据控制语言，用于系统角色控制和授权。通常是对用户和角色权限的授权和取消授权。由于用户和权限有较强的平台关联性和独特性，且主要由管理员编写管理，在实际应用系统编程时用的较少，故本专利将不涉及这部分。

Hive使用HiveQL进行数据操作,与Oracle SQL类似,HiveQL也有DDL和DML，但需要注意的是，Hive并不支持update和单条数据的insert和delete操作，Hive表中的数据或者是通过load语句初始化载入，或者是插入select 子查询。

与Oracle SQL对应的，HBase通过一系列自定义的Shell语句实现，其大致分类也包括：

1. DDL：主要用于表的建立、删除，列族的添加、删除、更新等。
2. DML：主要用于表数据的增删改查。

本专利设计的转换主要包括两部分：

1. 从Oracle SQL到HBase Shell的转换
2. 从Oracle SQL到HiveQL的转换

下面将详细介绍着两部分转换的过程和方法。

1. Oracle SQL转换成HBase Shell
   1. DDL转换

**建表：**

Oracle 建表语句如下：

CREATE TABLE tab1(col1 INTEGER,col2 VARCHAR2(20));

与之对应的HBase Shell如下：

create ‘tab1’,’col1’,’col2’;

**分析说明：**因为HBase的数据类型是单一的，只有字符串类型，所以建表的转换不涉及数据类型的映射，只需要将操作符进行转换，而像表名、列（族）名这样的对象标识基本不用变化。

**删表：**

Oracle删表语句如下：

DROP TABLE tab1;

与之对应的HBase Shell如下：

disable ‘tab1’

drop ‘tab1’

**分析说明：**可以看到了，在HBase中，删除一个表之前，需要先将该表失效。

**添加列（族）：**

Oracle添加列的语句如下：

ALTER TABLE tab1 ADD col3 INTEGER;

与之对应的HBase Shell如下：

disable ‘tab1’

alter ‘tab1’,{NAME => ‘col3’ }

enable tab1’

**分析说明：**可以看到了，在HBase中，为一个表添加列族之前，需要先将该表失效，在添加完成后，需要将该表重新生效。

**删除列（族）：**

Oracle删除列的语句如下：

ALTER TABLE tab1 DROP col3;

与之对应的HBase Shell如下：

disable ‘tab1’

alter 'tab1', NAME => 'col3', METHOD => 'delete'

enable tab1’

**分析说明：**可以看到了，在HBase中，从一个表中删除列族前后，同样需要对该表进行生失效操作。

* 1. DML转换

目前HBase可支持查询，对单条记录的插入以及单条记录删除和全表删除。

**查询：**

常见的Oracle查询语句如下：

SELECT col1, col2 FROM tab1;

SELECT COUNT(\*) FROM tab1;

SELECT col1, col2 FROM tab1 WHERE rowkey=’row1’;

SELECT col1, col2 FROM tab1 WHERE col1<3;

HBase不直接支持第四种查询方式，前三种查询方式对应的HBase Shell如下：

scan ‘tab1’

count ‘tab1’

get 'tab1', 'row1'

**分析说明：**HBase中全表查询的语句可以直接从Oracle SQL转换得到，基于Row Key的查询需要在Oracle表中有与Row Key对应的列，而按照特定查询条件，需要比较复杂的机制（Filter），超出单纯Shell语言的范畴，本专利不涉及。

**插入数据：**

Oracle向表插入数据的常用语句如下：

INSERT INTO tab1 VALUES(1,’a’);

INSERT INTO tab1 SELECT col1, col2 FROM tab2;

HBase Shell中目前只对单条记录插入，其语句如下：

put 'tab1', 'row1', 'col1:','1','col2:', 'a'

**分析说明：**可以看到，在HBase中，对单条记录的插入语句和Oracle相似，但没有语句直接支持通过子查询批量插入，实际应用中可将Hive和HBase整合实现此类功能，本专利对此不做赘述。

**删除数据：**

Oracle从表中删除数据的常用语句如下：

DELETE FROM tab1;

DELETE FROM tab1 where rowkey=‘row1’;

与之对应的HBase Shell语句如下：

truncate ‘tab1’

deleteall ‘tab1’,’row1’

**分析说明：**在HBase中，对单行记录的删除也是根据Row key的，尚不直接支持根据特定字段的删除，而全表的删除基本上和Oracle大同小异。

**更新数据：**

Oracle更新表中记录的语句如下：

UPDATE tab1 SET col1=2 WHERE rowkey=‘row1’;

对应的HBase Shell如下：

put ‘tab1’, ‘row1’,‘col1:’,‘2’

**分析说明：**可以看到，在HBase Shell中并没有专门的数据更新命令，数据更新是直接通过put重写。

1. Oracle SQL转换成HiveQL

和HBase不同，Hive中表的数据有不同的类型，因此，转换时需要考虑数据类型的映射，Hive基本数据类型如表1。

|  |  |  |
| --- | --- | --- |
| **数据类型** | **所占字节** | **支持版本** |
| TINYINT | 1byte，-128 ~ 127 |  |
| SMALLINT | 2byte，-32,768 ~ 32,767 |  |
| INT | 4byte,-2,147,483,648 ~ 2,147,483,647 |  |
| BIGINT | 8byte,-9,223,372,036,854,775,808 ~ 9,223,372,036,854,775,807 |  |
| BOOLEAN | TRUE或FALSE |  |
| FLOAT | 4byte单精度 |  |
| DOUBLE | 8byte双精度 |  |
| STRING | 字符串 |  |
| BINARY | 字节数组 | 从Hive0.8.0开始支持 |
| TIMESTAMP | 可为整型、浮点型或字符串 | 从Hive0.8.0开始支持 |
| DECIMAL |  | 从Hive0.11.0开始支持 |
| CHAR |  | 从Hive0.13.0开始支持 |
| VARCHAR |  | 从Hive0.12.0开始支持 |
| DATE |  | 从Hive0.12.0开始支持 |

表1

Oracle的主要数据类型如表2。

|  |  |  |
| --- | --- | --- |
| **数据类型** | **限制条件** | **说明** |
| CHAR | 最大长度2000 bytes | 固定长度字符串，可做索引的最大长度749 |
| VARCHAR2 | 最大长度4000 bytes | 可变长度的字符串 |
| NCHAR | 最大长度2000 bytes | 根据字符集而定的固定长度字符串 |
| NVARCHAR2 | 最大长度4000 bytes | 根据字符集而定的可变长度字符串 |
| DATE | DD-MM-YY（HH-MI-SS） | 日期（日-月-年） |
| LONG | 最大长度2G（231-1） | 超长字符串 |
| RAW | 最大长度2000 bytes | 固定长度的二进制数据，可存放多媒体图象声音等 |
| LONG RAW | 最大长度2G | 可变长度的二进制数据 |
| BLOB | 最大长度4G | 二进制数据 |
| CLOB | 最大长度4G | 字符数据 |
| NCLOB | 最大长度4G | 根据字符集而定的字符数据 |
| BFILE | 最大长度4G | 存放在数据库外的二进制数据 |
| NUMBER(P,S) | P为整数位，S为小数位 | 数字类型 |
| DECIMAL(P,S) | P为整数位，S为小数位 | 数字类型 |
| INTEGER | 小的整数 | 整数类型 |
| FLOAT | NUMBER(38)，双精度 | 浮点数类型 |
| REAL | NUMBER(63)，精度更高 | 实数类型 |

表2

本专利中，设定Oracle常用数据类型与Hive数据类型的的映射关系如表3。

|  |  |
| --- | --- |
| **Oracle** | **Hive** |
| CHAR | CHAR |
| VARCHAR2 | STRING |
| NCHAR | STRING |
| NVARCHAR2 | STRING |
| DATE | DATE |
| INTEGER | INT |
| FLOAT | FLOAT |
| REAL | DOUBLE |
| DECIMAL(P,S) | DECIMAL |
| NUMBER(P,S) | DECIMAL |

表3

* 1. DDL转换

**建表：**

Oracle 建表语句如下：

CREATE TABLE tab1(col1 INTEGER,col2 VARCHAR2(20));

与之对应的HiveQL如下：

CREATE TABLE tab1(col1 INT,col2 STRING);

**分析说明：**可以看到，除了类型略有不同，HiveQL的基本建表语句和Oracle SQL没有什么不同。

**删表：**

Oracle删表语句如下：

DROP TABLE tab1;

与之对应的HiveQL如下：

DROP TABLE tab1;

**分析说明：**在删除表时，Oracle SQL和HiveQL使用的是相同的语句。需要注意的是，Hive中的表有内部表和外部表之分，内部表删除时，表的元数据和数据会被一起删除，外部表删除时，只删除元数据。

**添加列：**

Oracle添加列的语句如下：

ALTER TABLE tab1 ADD (col3 INTEGER, col4 FLOAT;

与之对应的HiveQL如下：

ALTER TABLE tab1 ADD COLUMNS (col3 INT, col4 FLOAT);

**分析说明：**用HiveQL为表添加一个或多个字段时， 其基本语法与Oracle SQL一致，只是在关键字上有所区别。

**删除列：**

Oracle删除列的语句如下：

ALTER TABLE tab1 DROP col3;

与之对应的HiveQL如下：

ALTER TABLE tab1 REPLACE COLUMNS (col1 INTEGER, col2 VARCHAR2 (20));

**分析说明：** HiveQL中删除列的方式是将原来所有的列的删除，代之以新的列，转换时，需要从数据字典取得表的信息。

* 1. DML转换

HiveQL提供与Oracle SQL基本对应的查询操作支持，不支持记录级的增删改。

**查询：**

常见的Oracle查询语句如下：

SELECT col1, col2 FROM tab1;

SELECT COUNT(\*) FROM tab1;

SELECT col1, col2 FROM tab1 WHERE col1<3;

Select col2,avg(col1) from tab1 group by col2;

与之对应的HiveQL如下：

SELECT col1, col2 FROM tab1;

SELECT COUNT(\*) FROM tab1;

SELECT col1, col2 FROM tab1 WHERE col1<3;

Select col2,avg(col1) from tab1 group by col2;

**分析说明：**可以看到，较之HiveQL支持更多的查询操作，其语法结构基本和Oracle SQL无区别。

**插入数据：**

Oracle向表插入数据的常用语句如下：

INSERT INTO tab1 VALUES(1,’a’);

INSERT INTO tab1 SELECT col1, col2 FROM tab2;

HiveQL中目前只支持通过子查询的方式插入数据，其语句如下：

INSERT INTO tab1 SELECT col1, col2 FROM tab2;

INSERT OVERWRITE tab1 SELECT col1, col2 FROM tab2;

**分析说明：**通过查询插入数据时，HiveQL的语法架构和Oracle SQL基本相同，但多了INSERT OVERWRITE功能，即用查询的数据覆盖原表中的数据，而INSERT INTO功能是在原表中追加数据。

**删除数据：**

Oracle从表中全表删除数据的语句如下：

DELETE FROM tab1;

**分析说明：**HiveQL中并没有直接删除表数据的语句，一般是通过删除表，然后再重建的方式。

**更新数据：**

Oracle更新表中记录的语句如下：

UPDATE tab1 SET col1=2 WHERE col1=‘1’;

**分析说明：**HiveQL中并没有直接支持数据更新的语句。

上文详细描述了本专利方案执行脚本语言转换的机制，可以看出，如果通过手工改写，其复杂度和工作量会很大。本专利针对此转换过程，建立了一套自动化、智能、准确的分析方法和装置，在获得Oracle SQL脚本输入后，整个分析过程无需人工干预，便可生成最终的目标框架（HBase，Hive）脚本，保证转换后的脚本可以在目标数据系统中无差异地执行。

下面详细描述本专利针对上述脚本分析转换方法建立的处理系统，本系统具有多层级结构，系统总体架构层次图如下图1。



图1 – 脚本语言转换系统架构层次图

用户接口层：负责和用户进行交互处理。本系统用户界面基于命令行方式，用户接口层的主要功能组件有：

* 命令处理：用户输入命令接收、解析与校验，并调用下层服务进行处理；
* 信息交互：即时将回退生成和回滚执行的过程与结果展示给用户；

分析转换层：系统核心层次，负责SQL脚本及语句解析，并根据映射规则规则生成目标语句及脚本。SQL解析根据不同的数据库类型采用不同的SQL引擎，每一种数据库类型分别对应一个SQL引擎，SQL引擎基于ANTLR工具完成。分析转换层的主要功能组件有：

* SQL脚本解析：调用语法分析引擎完成用户SQL脚本解析，并负责解析过程的错误处理；
* 转换分析：完成解析源SQL语句后，分析解析结果，通过元数据管理模块获得目标脚本元数据（包括关键字、语法模板、数据类型映射关系等），同时，通过数据服务层获取SQL语句中涉及的数据对象（本专利中特指表）的数据字典信息；
* 目标脚本生成：根据转换分析模块的处理结果，完成目标脚本的生成；
* 语法分析引擎：根据具体类型的数据库SQL语句词法和语法规则，完成SQL语句解析的组件。采用ANTLR进行词法解析和语法解析，ANTLR是一个开源的语言识别工具，包括词法分析器（Lexer）和语法分析器（Parser）。

数据服务层：主要功能是从数据库获得SQL基本中各数据表的数据字典信息。

本系统的组件图如下图2，描述了主要系统组件，及各组件之间的交互。



图2 – 脚本转换系统组件图

综合上图，本专利方案对Oracle脚本转换处理的步骤如下：

1）获得待转换SQL脚本输入和数据库授权

执行回退处理首先需要获得待转换脚本的路径。待转换脚本指的是包含SQL语句的文本文件，可以是一个或多个。

此外还必须获得数据库的操作授权，通常指的是对应的数据库用户名和密码。本方案要求的数据库权限需要等于或大于待转换脚本执行用户的数据库权限。

2）对源SQL脚本执行解析，生成语句解析结果

获得源SQL脚本后，首先需要对脚本进行SQL解析。本专利方案采用ANTLR词法分析和语法分析进行SQL引擎实现。

传统的语言解析方法，是首先按照规则，将文本中的字符串划分为一个个符号（Token），Token主要有两种属性，Token名称（Token Name），Token值（Token Value），对整个文件的不断处理会连续的输出Token流，整个过程称之为词法分析。词法分析概括起来说，就是定义Token类型和划分规则，并执行分析获得Token流的过程。

语法分析是根据定义的语法规则，获得输入的Token流，进行语法树构建，最终识别语句的过程。

本专利方案支持Oracle大部分常见语句，且可以通过扩展语法分析引擎，支持新的数据库类型，扩展词法与语法规则，支持新的语句类型。

下面以SQL语句中常见的INSERT语句为例，介绍本专利方案语法分析的工作方法和流程。

a）词法分析

假设要向数据表tab1中插入一行数据(1,’a’)，存在如下两种写法，且都是合法的INSERT语句。

INSERT INTO tab1 VALUES(1,’a’);

INSERT INTO tab1 (col1, col2) VALUES (1,’a’);

在上述语句中，可以发现如下规律：

* INSERT语句必须以INSERT INTO关键字开始；
* 接下来必须跟上待插入数据的表名，此处假设表名只能是字母、数字、下划线的组合，且必须以字母开头；
* 表名后可以包含列名，也可以不包含列名。不包含列名表示所有列，包含列名则必须以左括号开始，右括号结束，列名以逗号分隔。此处同样假设列名只能是字母、数字、下划线的组合，且必须以字母开头；
* 之后是VALUES子句；
* VALUES子句后，是具体的待插入值，以左括号开始，右括号结束，多个值以逗号分隔，如果值为字符串，以引号开始，引号结束。

根据如上分析结果，可以总结如下INSERT语句的Token类型。

|  |  |
| --- | --- |
| **Token类型** | **匹配规则** |
| INSERT | INSERT，不区分大小写，也可以是insert,Insert等 |
| INTO | INTO，不区分大小写 |
| ( | 左括号 |
| ) | 右括号 |
| , | 逗号 |
| VALUES | VALUES，不区分大小写 |
| STRING | 字符串，以'开始，以'结束包含的非引号非回车字符串 |
| ; | SQL语句分隔符 |
| NAME | 标识符，此处指的是表名、列名，只能是字母、数字、下划线的组合，且必须以字母开头 |
| NUMBER | 数字，0-9的组合 |

ANTLR使用基于正则表达式的词法匹配规则，那么此处针对如上INSERT语句，定义的ANTLR词法文件如下：

[Ii][Nn][Ss][Ee][Rr][Tt] { return INSERT; }

[Ii][Nn][Tt][Oo] { return INTO; }

[Vv][Aa][Ll][Uu][Ee][Ss] { return VALUES; }

"(" | ")" | "," | ";" { return yytext[0]; }

[A-Za-z][A-Za-z0-9\_]\* { return NAME; }

[0-9]+ { return NUMBER; }

"'"[^"'"\n]\*"'" { return STRING; }

[ \t]

. { yyerror("匹配失败"); }

根据如上词法文件，对上述第一条INSERT语句的词法分析结果如下表。

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 语句 | INSERT | INTO | tab1 | VALUES | ( | 1 | , | 'a ' | ) | ; |
| Token | INSERT | INTO | NAME | VALUES | ( | NUMBER | , | STRING | ) | ; |

b）语法分析

不同的语言有不同的语法，SQL语言也是如此。就SQL语言来说，虽然存在SQL国际标准，但是不同的数据库厂商对自身的数据库产品往往进行了一些增强和扩展，所以对不同的数据库类型，语法也具有一些差异。进行语法分析前，首先要确定语言的语法规则，然后才能根据规则分析输入语句。

在ANTLR中，对语法规则的定义是主要是通过词法分析输出的Token来完成的。Token是语法规则的基本元素，在语法分析的范畴内，Token被称为终结符，通过终结符建立最终的有语义的语句。同样以上述INSERT语句为例，说明语法解析的方法和过程。

首先，根据SQL语句格式定义如下INSERT语法规则：

stmt\_list: stmt ';'

| stmt\_list ';'

;

stmt: insert\_stmt

| delete\_stmt

| update\_stmt

| select\_stmt

;

insert\_stmt: INSERT INTO NAME column\_list VALUES '(' value\_list ')'

;

column\_list: /\* null \*/ | '(' field\_list ')';

;

field\_list: NAME | ',' field\_list

;

value\_list: S | NUMBER | ',' value\_list

;

根据上述语法规则，对语句“INSERT INTO tab1 (col1, col2) VALUES (1, 'a ');”的语法解析过程如下图3.

;

INSERT

INTO

NAME

VALUES

(

)

(

)

NAME

NAME

,

NUMBER

,

STRING

图3 – INSERT语句语法树

上图展示了标准INSERT语句语法解析的最终语法树，方框内的叶子节点为终结符，终结符均是词法分析的输出，树上的椭圆形节点均为非终结符，树的结构则展示了语句的语法规则。不仅是INSERT语句，其他的任何SQL语句都可以通过定义语法规则来完成语义解析。

3）根据转换映射规则和解析结果，生成回退目标脚本语句

完成SQL语句解析后，接下来需要根据SQL语句分析出目标脚本语句，即进行转换映射的过程。

前文已详细介绍了Oracle SQL语句的到HBase和Hive语句的转换方法，此处主要描述针对转换映射规则的实现，同样以上述INSERT语句为例，结合前文对INSERT语句的转换映射分析方法，可得到系统对INSERT语句进行转换分析的流程图，如下图4。



图4 – INSERT语句转换分析流程图

根据处理流程图和语法解析的结果，对上述INSERT语句向HBase的转换分析过程如下表所示。

|  |  |  |  |
| --- | --- | --- | --- |
| **Token名** | **Token值** | **转换分析过程** | **目标脚本转换输出** |
| INSERT | ‘INSERT’ | 1、由INSERT可知将生成put语句； | put |
| INTO | ‘INTO’ |  |  |
| NAME | ‘tab1’ | 1. 获得表名 2. 查询主键或唯一索引，此处假设为col1； 3. 根据主键或唯一索引生成Hbase RowKey； 4. 查询表全部字段列表，为col1, col2; | 'tab1', $ROWKEY, |
| ( | ‘(’ |  |  |
| NAME | ‘col1’ |  | col1 =  'col1:' $STRING, |
| , | ‘,’ |  | , |
| NAME | ‘col2’ |  | 'col2:', $STRING |
| ) | ‘)’ |  |  |
| VALUES | ‘VALUES’ |  |  |
| ( | ‘(’ |  |  |
| NUMBER | 1 |  | $STRING = ‘1’ |
| , | ‘,’ |  |  |
| STRING | ‘test’ |  | $NUMBER = ‘a’ |
| ) | ‘)’ | 1. 获得字段待插入值,分别为1, ‘a’; 2. 获得主键col1值 | $ROWKEY ='1' |

上表Token名值两列标识输入的Token类型和值，转换分析过程展示了分析的逻辑处理，目标脚本转换逻辑输出列显示了逆向分析的结果。不光是INSERT语句，其他SQL语句同样适合类似的分析方法和系统。

4）将转换逻辑语句转换为可执行的目标脚本

源SQL语句经过转换分析后，得到的逻辑语句通常为子句的序列，因此还需要进一步转换，生成可执行的回退语句。本阶段进行的主要工作有：

a） 重组。转换分析的结果为一系列转换脚本语句的子句，这些子句是按照源SQL语句的结构分析输出的，而目标脚本语句的结构通常与源SQL语句不同，因此需要对逻辑子句进行重新组织。

b）映射。前文提到，HBase和Hive有不同的语法，同样的Oracle SQL语句在不同的目标系统里有不同的实现，系统需要将具有不同实现的语句映射为目标系统的语句。

d）合并。经过重组、映射后的目标脚本语句，需要合并为可执行的脚本文件，合并负责语句和命令的组织，以及脚本文件生成。

如上三个步骤为目标脚本语句生成的过程，某些过程可能是可选的，实际处理过程中，会根据语句的特征进行特定的处理。如下示例展示了INSERT语句目标脚本语句生成的过程。

|  |  |
| --- | --- |
| **过程** | **输出** |
| 转换分析 | put  'tab1',  $ROWKEY,  'col1:',$STRING,  'col2:', $STRING  $STRING=1  $STRING=’a’  $ROWKEY=’1’ |
| 重组 | put  'tab1'  ‘1’  'col1:', ‘1’  'col2:', ‘a’ |
| 映射 | put 'tab1', ‘1’, 'col1:', ‘1’,'col2:', ‘a’ |
| 合并 | 目标脚本文件 |

本专利方案实施脚本转换的方法简单，在目标系统中执行转换脚本即可。

六、本申请提案的关键点和欲保护点

1）针对HBase和Hive，提出了将Oracle SQL语句进行转换分析的方法和过程。

2）针对提出的转换分析方法和过程，建立了一套支持HBase和Hive的脚本生成系统，并可通过扩展支持新的源数据库系统和目标数据库系统。

**注：提炼出了本方案中作出创新性改进的重要技术点，并按照重要性顺序逐一列出。**

七、与第三条中最接近的现有技术相比，本申请提案有何技术优点

本发明提出的方法和系统，具有以下有益效果：首先是实用性高，目前支持业界主流的Oracle数据库系统向主流大数据平台HBase和Hive的转换，通过扩展语法分析引擎，还可以支持新的源数据库类型和语法，增加了各异构数据平台之间的访问透明度，提供了无缝的数据整合和系统升级方案；其次是正确性高，系统采用自动化处理方法，基于比较成熟的ANTLR语法分析工具，具有良好的错误处理机制，同时避免了人为错误；再次是操作简单，本方法和系统对用户来说，输入输出简单，复杂的分析处理过程全部在系统内部完成，对用户没有技术要求；最后是成本低，本方法和系统在进行转换分析和生成目标脚本时都是独立运行，不会影响其他系统功能，且自动化处理效率高，减少了人力需求和资源需求，节约了成本。

**注：对应于第四部分中本发明要解决的技术问题，逐一写明了在解决技术问题之后取得的技术效果。**

八、其他有助于理解本申请提案的技术资料

1、Oracle Database Online Documentation 10g Release 2(10.2).

http://www.oracle.com/pls/db102/homepage

2、Pragmatic.The.Definitive.ANTLR.4.Reference, Terence Parr 著，

The Pragmatic Bookshelf出版社；

3、HBase the definitive guide，Lars George著，O’RELLY出版社；

4、Hadoop：The.Definitive.Guide(3rd,2012.5)，Tom.White著，O’RELLY

出版社；

**注：该标准与本发明密切相关，且对理解本发明有帮助。**

**提醒：本技术交底书示例所述技术方案尚未公开，请注意保密！**