云数据迁移

常见问题

文档版本 2

发布日期 2020-08-25

华为技术有限公司

商标声明


和其他华为商标均为华为技术有限公司的商标。
本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受华为公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，华为公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

目 录

1 通用类

1.1 CDM 有哪些优势?

云数据迁移（Cloud Data Migration，简称CDM）服务基于分布式计算框架，利用并行化处理技术，使用CDM迁移数据的优势如表1-1所示。

表 1-1 CDM 优势

优势项	用户自行开发	CDM
易使用	自行准备服务器资源，安装配置必要的软件并进行配置，等待时间长。 程序在读写两端会根据数据源类型，使用不同的访问接口，一般是数据源提供的对外接口，例如JDBC、原生API等，因此在开发脚本时需要依赖大量的库、SDK等，开发管理成本较高。	CDM提供了Web化的管理控制台，通过Web页实时开通服务。 用户只需要通过可视化界面对数据源和迁移任务进行配置，服务会对数据源和任务进行全面的管理和维护，用户只需关注数据迁移的具体逻辑，而不用关心环境等问题，极大降低了开发维护成本。 CDM还提供了REST API，支持第三方系统调用和集成。
实时监控	需要自行选型开发。	您可以使用云监控服务监控您的CDM集群，执行自动实时监控、告警和通知操作，帮助您更好地了解CDM集群的各项性能指标。
免运维	需要自行开发完善运维功能，自行保证系统可用性，尤其是告警及通知功能，否则只能人工值守。	使用CDM服务，用户不需要维护服务器、虚拟机等资源。CDM的日志，监控和告警功能，有异常可以及时通知相关人员，避免7*24小时人工值守。

优势项	用户自行开发	CDM
高效率	在迁移过程中，数据读写过程都是由一个单一任务完成的，受限于资源，整体性能较低，对于海量数据场景往往不能满足要求。	CDM任务基于分布式计算框架，自动将任务切分为独立的子任务并行执行，能够极大提高数据迁移的效率。针对Hive、HBase、MySQL、DWS（数据仓库服务）数据源，使用高效的数据导入接口导入数据。
多种数据源支持	数据源类型繁杂，针对不同数据源开发不同的任务，脚本数量成千上万。	支持数据库、Hadoop、NoSQL、数据仓库、文件等多种类型的数据源。
多种网络环境支持	随着云计算技术的发展，用户数据可能存在于各种环境中，例如公有云、自建/托管IDC、混合场景等。在异构环境中进行数据迁移需要考虑网络连通性等因素，给开发和维护都带来较大难度。	无论数据是在用户本地自建的IDC中（Internet Data Center，互联网数据中心）、云服务中、第三方云中，或者使用ECS自建的数据库或文件系统中，CDM均可帮助用户轻松应对各种数据迁移场景，包括数据上云，云上数据交换，以及云上数据回流本地业务系统。

1.2 CDM 可迁移哪些业务数据?

云数据迁移（Cloud Data Migration，以下简称CDM）提供同构/异构数据源之间批量数据迁移服务，帮助您实现数据自由流动。支持自建和云上的文件系统，关系数据库，数据仓库，NoSQL，大数据云服务，对象存储等数据源。

CDM支持两种迁移方式：

- 表/文件迁移：适用于数据上云、云服务间数据交换、云上数据回流到本地业务系统。
- 整库迁移：适用于数据库上云。

表/文件迁移时支持的数据源如**表1-2**所示。

表 1-2 表/文件迁移支持的数据源

数据源分类	数据源	导出	导入
数据仓库	数据仓库服务（DWS）	支持	支持
	数据湖探索（DLI）	不支持	支持
	FusionInsight LibrA	支持	支持
Hadoop	MRS HDFS	支持	支持
	MRS HBase	支持	支持
	MRS Hive	支持	支持
	FusionInsight HDFS	支持	支持

数据源分类	数据源	导出	导入
	Apache HDFS	支持	支持
	Hadoop发行版	不支持	不支持
	FusionInsight HBase	支持	支持
	FusionInsight Hive	支持	支持
	Apache HBase	支持	支持
	Apache Hive	支持	支持
对象存储	对象存储服务（OBS）	支持	支持
	阿里云对象存储（OSS）	支持	不支持
	七牛云对象存储（KODO）	支持	不支持
	亚马逊对象存储S3	支持	不支持
	腾讯云对象存储（COS）	支持	不支持
文件系统	FTP	支持	支持
	SFTP	支持	支持
	HTTP	支持	不支持
	网络附加存储（NAS）	支持	支持
	弹性文件服务（SFS Turbo）	支持	支持
关系数据库	云数据库 MySQL	支持	支持
	云数据库 PostgreSQL	支持	支持
	云数据库 SQL Server	支持	支持
	分布式数据库中间件（DDM）	支持	不支持
	MySQL	支持	支持
	PostgreSQL	支持	支持
	Microsoft SQL Server	支持	支持
	Oracle	支持	支持
	IBM Db2	支持	支持
	Derecho（GaussDB）	支持	支持
	NewSQL (GaussDB)	支持	支持
	SAP HANA	支持	支持
	MYCAT	支持	支持

数据源分类	数据源	导出	导入
NoSQL	分布式缓存服务（DCS）	不支持	支持
	文档数据库服务（DDS）	支持	支持
	表格存储服务（CloudTable）	支持	支持
	CloudTable OpenTSDB	支持	支持
	Redis	支持	支持
	MongoDB	支持	支持
	Cassandra	支持	支持
搜索	云搜索服务（CSS）	支持	支持
	Elasticsearch	支持	支持
消息系统	数据接入服务（DIS）	支持	支持
	Apache Kafka	支持	支持
	MRS Kafka	支持	支持
	DMS Kafka	支持	支持

说明

- 上表中非云服务的数据源，例如MySQL，既可以支持用户本地数据中心自建的MySQL，也可以是用户在ECS上自建的MySQL，还可以是第三方云的MySQL服务。
- 消息系统（DIS/Apache Kafka/DMS Kafka）作为源端时，CDM仅支持迁移到以下类型的数据源：
  - 云搜索服务（CSS）
  - 数据接入服务（DIS）
  - Apache Kafka
  - DMS Kafka

批量数据迁移支持整库迁移的数据源如表1-3所示。

表 1-3 批量数据迁移支持整库迁移的数据源

源端数据源类型	目的端数据源类型					
	RDS for MySQL	MRS(Hive)	DWS	CSS	OBS	CloudTable
MySQL	√	√	√	×	√	×
PostgreSQL	√	√	√	×	√	×
Microsoft SQL Server	√	√	√	×	×	×

Oracle	√	√	√	×	√	×
Elasticsearch	×	×	×	√	×	×
MongoDB	×	×	×	×	×	×
HBase	×	×	×	×	×	√
IBM Db2	√	√	√	×	√	×
Derecho(Gauss DB)	√	√	√	×	√	×
DDM	√	√	√	×	√	×
SAP HANA	√	√	√	×	√	×
DWS	√	√	√	×	×	×
Hive	√	×	√	×	×	×

1.3 CDM 有哪些安全防护?

CDM是一个完全托管的服务，提供了以下安全防护能力保护用户数据安全。

- 实例隔离：CDM服务的用户只能使用自己创建的实例，实例和实例之间是相互隔离的，不可相互访问。
- 系统加固：CDM实例的操作系统进行了特别的安全加固，攻击者无法从Internet访问CDM实例的操作系统。
- 密钥加密：用户在CDM上创建连接输入的各种数据源的密钥，CDM均采用高强度加密算法保存在CDM数据库。
- 无中间存储：数据在迁移的过程中，CDM只处理数据映射和转换，而不会存储任何用户数据或片段。

1.4 CDM 迁移性能如何?

单个CDM实例理论上可以支持1TB～8TB/天的数据迁移，取决于网络带宽和数据源的读写性能。如果给不同的业务部门使用，例如财务、网上商城，建议使用不同的CDM实例。

1.5 如何降低成本?

1. 如果每天只是特定时间段导数据，可以使用CDM关机功能，平常关机，仅需要迁移数据时才开机。由于关机后仅收取五分钱一小时实例资源占用费，一天最低只需要1.2元。
2. 如果是迁移公网的数据上云，可以使用华为云**NAT网关**服务，实现CDM服务与子网中的其他弹性云服务器共享弹性IP，可以更经济、更方便的通过Internet迁移本地数据中心或第三方云上的数据。

   具体操作如下：

a. 假设已经创建好了CDM集群（无需为CDM集群绑定专用弹性IP），记录下CDM集群所在的VPC和子网。

b. 创建NAT网关，注意选择和CDM集群相同的VPC、子网。

c. 创建完NAT网关后，回到NAT网关控制台列表，单击创建好的网关名称，然后选择“添加SNAT规则”。

图 1-1 添加 SNAT 规则

d. 选择子网和弹性IP，如果没有弹性IP，需要先申请一个。

完成之后，就可以到CDM控制台，通过Internet迁移公网的数据上云了。例如：迁移本地数据中心FTP服务器上的文件到OBS、迁移第三方云上关系型数据库到云服务RDS。

说明

如果用户对本地数据源的访问通道做了SSL加密，则CDM无法通过弹性IP连接数据源。

2 功能类

2.1 是否支持增量迁移?

CDM支持增量数据迁移。利用定时任务配置和时间宏变量函数等参数，可支持以下场景的增量数据迁移：

- **文件增量迁移**
- **关系数据库增量迁移**
- **HBase/CloudTable增量迁移**
- **使用时间宏变量完成增量同步**

2.2 是否支持字段转换?

支持，CDM服务支持以下字段转换器：

- **脱敏**
- **去前后空格**
- **字符串反转**
- **字符串替换**
- **表达式转换**

在创建表/文件迁移作业的字段映射界面，可新建字段转换器，如**图2-1**所示。

图 2-1 新建字段转换器

脱敏

隐藏字符串中的关键信息，例如要将“12345678910”转换为“123****8910”，则配置如下：

- “起始保留长度”为“3”。
- “结尾保留长度”为“4”。
- “替换字符”为“*”。

图 2-2 字段脱敏

去前后空格

自动去字符串前后的空值，不需要配置参数。

字符串反转

自动反转字符串，例如将“ABC”转换为“CBA”，不需要配置参数。

字符串替换

替换字符串，需要用户配置被替换的对象，以及替换后的值。

表达式转换

使用JSP表达式语言（Expression Language）对当前字段或整行数据进行转换。JSP表达式语言可以用来创建算术和逻辑表达式。在表达式内可以使用整型数，浮点数，字符串，常量true、false和null。

表达式支持以下两个环境变量：

- value：当前字段值。
- row：当前行，数组类型。

表达式支持以下工具类：

- StringUtils：字符串处理类，参考Java SDK代码的包结构“org.apache.commons.lang.StringUtils”。
- DateUtils：日期工具类。
- CommonUtils：公共工具类。
- NumberUtils：字符串转数值类。

- HttpsUtils：读取网络文件类。

应用举例：

1. 给当前字段设置一个字符串常量，例如设置为VIP。
   表达式："*VIP*"
2. 如果当前字段为字符串类型，将字符串全部转换为小写，例如将“aBC”转换为“abc”。
   表达式：StringUtils.lowerCase(value)
3. 将当前字段的字符串全部转为大写。
   表达式：StringUtils.upperCase(value)
4. 如果当前字段值为“yyyy-MM-dd”格式的日期字符串，需要截取年，例如字段值为“2017-12-01”，转换后为“2017”。
   表达式：StringUtils.substringBefore(value,"-")
5. 如果当前字段值为数值类型，转换后值为当前值的两倍。
   表达式：value*2
6. 如果当前字段值为“true”，转换后为“Y”，其它值则转换后为“N”。
   表达式：value=="true"?"Y":"N"
7. 如果当前字段值为字符串类型，当为空时，转换为“Default”，否则不转换。
   表达式：empty value? "Default":value
8. 如果当前字段、第1、第2字段为数值类型，当前字段转换为第1字段与第2字段拼接后的值。
   表达式：row[0]+row[1]
9. 如果当前字段为Date或Timestamp类型，转换后返回当前年份，类型为Int。
   表达式：DateUtils.getYear(value)
10. 如果当前字段为“yyyy-MM-dd”格式的时间日期字符串，转换成Date类型。
    表达式：DateUtils.format(value,"yyyy-MM-dd")
11. 如果想将日期字段格式从“2018/01/05 15:15:05”转换为“2018-01-05 15:15:05”。
    表达式：DateUtils.format(DateUtils.parseDate(value,"yyyy/MM/dd HH:mm:ss"),"yyyy-MM-dd HH:mm:ss")
12. 获取一个36位的UUID（Universally Unique Identifier，通用唯一识别码）。
    表达式：CommonUtils.randomUUID()
13. 如果当前字段值为字符串类型，将首字母转换为大写，例如将“cat”转换为“Cat”。
    表达式：StringUtils.capitalize(value)
14. 如果当前字段值为字符串类型，将首字母转换为小写，例如将“Cat”转换为“cat”。
    表达式：StringUtils.uncapitalize(value)
15. 如果当前字段值为字符串类型，使用空格填充为指定长度，并且将字符串居中，当字符串长度不小于指定长度时不转换，例如将“ab”转换为长度为4的“ab”。
    表达式：StringUtils.center(value,*4*)
16. 删除字符串末尾的一个换行符（包括“\n”、“\r”或者“\r\n”），例如将“abc\r\n\r\n”转换为“abc\r\n”。

表达式：StringUtils.chomp(value)

17. 如果字符串中包含指定的字符串，则返回布尔值true，否则返回false。例如“abc”中包含“a”，则返回true。

    表达式：StringUtils.contains(value,"*a*")

18. 如果字符串中包含指定字符串的任一字符，则返回布尔值true，否则返回false。例如“zzabyycdxx”中包含“z”或“a”任意一个，则返回true。

    表达式：StringUtils.containsAny("value","*za*")

19. 如果字符串中不包含指定的所有字符，则返回布尔值true，包含任意一个字符则返回false。例如“abz”中包含“xyz”里的任意一个字符，则返回false。

    表达式：StringUtils.containsNone(value,"*xyz*")

20. 如果当前字符串只包含指定字符串中的字符，则返回布尔值true，包含任意一个其它字符则返回false。例如“abab”只包含“abc”中的字符，则返回true。

    表达式：StringUtils.containsOnly(value,"*abc*")

21. 如果字符串为空或null，则转换为指定的字符串，否则不转换。例如将空字符转换为null。

    表达式：StringUtils.defaultIfEmpty(value,*null*)

22. 如果字符串以指定的后缀结尾（包括大小写），则返回布尔值true，否则返回false。例如“abcdef”后缀不为null，则返回false。

    表达式：StringUtils.endsWith(value,*null*)

23. 如果字符串和指定的字符串完全一样（包括大小写），则返回布尔值true，否则返回false。例如比较字符串“abc”和“ABC”，则返回false。

    表达式：StringUtils.equals(value,"*ABC*")

24. 从字符串中获取指定字符串的第一个索引，没有则返回整数-1。例如从“aabaabaa”中获取“ab”的第一个索引1。

    表达式：StringUtils.indexOf(value,"*ab*")

25. 从字符串中获取指定字符串的最后一个索引，没有则返回整数-1。例如从“aFkyk”中获取“k”的最后一个索引4。

    表达式：StringUtils.lastIndexOf(value,"*k*")

26. 从字符串中指定的位置往后查找，获取指定字符串的第一个索引，没有则转换为“-1”。例如“aabaabaa”中索引3的后面，第一个“b”的索引是5。

    表达式：StringUtils.indexOf(value,"*b*",*3*)

27. 从字符串获取指定字符串中任一字符的第一个索引，没有则返回整数-1。例如从“zzabyycdxx”中获取“z”或“a”的第一个索引0。

    表达式：StringUtils.indexOfAny(value,"*za*")

28. 如果字符串仅包含Unicode字符，返回布尔值true，否则返回false。例如“ab2c”中包含非Unicode字符，返回false。

    表达式：StringUtils.isAlpha(value)

29. 如果字符串仅包含Unicode字符或数字，返回布尔值true，否则返回false。例如“ab2c”中仅包含Unicode字符和数字，返回true。

    表达式：StringUtils.isAlphanumeric(value)

30. 如果字符串仅包含Unicode字符、数字或空格，返回布尔值true，否则返回false。例如“ab2c”中仅包含Unicode字符和数字，返回true。

    表达式：StringUtils.isAlphanumericSpace(value)

31. 如果字符串仅包含Unicode字符或空格，返回布尔值true，否则返回false。例如“ab2c”中包含Unicode字符和数字，返回false。

表达式：StringUtils.isAlphaSpace(value)

32. 如果字符串仅包含ASCII可打印字符，返回布尔值true，否则返回false。例如“! ab-c~”返回true。

    表达式：StringUtils.isAsciiPrintable(value)

33. 如果字符串为空或null，返回布尔值true，否则返回false。

    表达式：StringUtils.isEmpty(value)

34. 如果字符串中仅包含Unicode数字，返回布尔值true，否则返回false。

    表达式：StringUtils.isNumeric(value)

35. 获取字符串最左端的指定长度的字符，例如获取“abc”最左端的2位字符“ab”。

    表达式：StringUtils.left(value,*2*)

36. 获取字符串最右端的指定长度的字符，例如获取“abc”最右端的2位字符“bc”。

    表达式：StringUtils.right(value,*2*)

37. 将指定字符串拼接至当前字符串的左侧，需同时指定拼接后的字符串长度，如果当前字符串长度不小于指定长度，则不转换。例如将“yz”拼接到“bat”左侧，拼接后长度为8，则转换后为“yzyzybat”。

    表达式：StringUtils.leftPad(value,*8*,"*yz*")

38. 将指定字符串拼接至当前字符串的右侧，需同时指定拼接后的字符串长度，如果当前字符串长度不小于指定长度，则不转换。例如将“yz”拼接到“bat”右侧，拼接后长度为8，则转换后为“batyzyzy”。

    表达式：StringUtils.rightPad(value,*8*,"*yz*")

39. 如果当前字段为字符串类型，获取当前字符串的长度，如果该字符串为null，则返回0。

    表达式：StringUtils.length(value)

40. 如果当前字段为字符串类型，删除其中所有的指定字符串，例如从“queued”中删除“ue”，转换后为“qd”。

    表达式：StringUtils.remove(value,"*ue*")

41. 如果当前字段为字符串类型，移除当前字段末尾指定的子字符串。指定的子字符串若不在当前字段的末尾，则不转换，例如移除当前字段“www.domain.com”后的“.com”。

    表达式：StringUtils.removeEnd(value,"*.com*")

42. 如果当前字段为字符串类型，移除当前字段开头指定的子字符串。指定的子字符串若不在当前字段的开头，则不转换，例如移除当前字段“www.domain.com”前的“www.”。

    表达式：StringUtils.removeStart(value,"*www.*")

43. 如果当前字段为字符串类型，替换当前字段中所有的指定字符串，例如将“aba”中的“a”用“z”替换，转换后为“zbz”。

    表达式：StringUtils.replace(value,"*a*","*z*")

44. 如果当前字段为字符串类型，一次替换字符串中的多个字符，例如将字符串“hello”中的“h”用“j”替换，“o”用“y”替换，转换后为“jelly”。

    表达式：StringUtils.replaceChars(value,"*ho*","*jy*")

45. 如果当前字段为字符串类型，使用指定分隔符将提供的文本拆分为数组。例如将“ab:cd:ef”按“:”分隔，转换后为“["ab","cd","ef"]”。

    表达式：StringUtils.split(value,":")

46. 如果字符串以指定的前缀开头（区分大小写），则返回布尔值true，否则返回false，例如当前字符串“abcdef”以“abc”开头，则返回true。

    表达式：StringUtils.startsWith(value,"*abc*")

47. 如果当前字段为字符串类型，去除字段中所有指定的字符，例如去除“abcyx”中所有的“x”、“y”和“z”，转换后为“abc”。

    表达式：StringUtils.strip(value,"*xyz*")

48. 如果当前字段为字符串类型，去除字段末尾所有指定的字符，例如去除当前字段末尾的所有空格。

    表达式：StringUtils.stripEnd(value,*null*)

49. 如果当前字段为字符串类型，去除字段开头所有指定的字符，例如去除当前字段开头的所有空格。

    表达式：StringUtils.stripStart(value,*null*)

50. 如果当前字段为字符串类型，获取字符串指定位置后（不包括指定位置的字符）的子字符串，指定位置如果为负数，则从末尾往前计算位置。例如获取“abcde”第2个字符后的字符串，则转换后为“cde”。

    表达式：StringUtils.substring(value,*2*)

51. 如果当前字段为字符串类型，获取字符串指定区间的子字符串，区间位置如果为负数，则从末尾往前计算位置。例如获取“abcde”第2个字符后、第5个字符前的字符串，则转换后为“cd”。

    表达式：StringUtils.substring(value,*2*,*5*)

52. 如果当前字段为字符串类型，获取当前字段里第一个指定字符后的子字符串。例如获取“abcba”中第一个“b”之后的子字符串，转换后为“cba”。

    表达式：StringUtils.substringAfter(value,"*b*")

53. 如果当前字段为字符串类型，获取当前字段里最后一个指定字符后的子字符串。例如获取“abcba”中最后一个“b”之后的子字符串，转换后为“a”。

    表达式：StringUtils.substringAfterLast(value,"*b*")

54. 如果当前字段为字符串类型，获取当前字段里第一个指定字符前的子字符串。例如获取“abcba”中第一个“b”之前的子字符串，转换后为“a”。

    表达式：StringUtils.substringBefore(value,"*b*")

55. 如果当前字段为字符串类型，获取当前字段里最后一个指定字符前的子字符串。例如获取“abcba”中最后一个“b”之前的子字符串，转换后为“abc”。

    表达式：StringUtils.substringBeforeLast(value,"*b*")

56. 如果当前字段为字符串类型，获取嵌套在指定字符串之间的子字符串，没有匹配的则返回null。例如获取“tagabctag”中“tag”之间的子字符串，转换后为“abc”。

    表达式：StringUtils.substringBetween(value,"*tag*")

57. 如果当前字段为字符串类型，删除当前字符串两端的控制字符（char≤32），例如删除字符串前后的空格。

    表达式：StringUtils.trim(value)

58. 将当前字符串转换为字节，如果转换失败，则返回0。

    表达式：NumberUtils.toByte(value)

59. 将当前字符串转换为字节，如果转换失败，则返回指定值，例如指定值配置为1。

    表达式：NumberUtils.toByte(value,*1*)

60. 将当前字符串转换为Double数值，如果转换失败，则返回0.0d。

表达式：NumberUtils.toDouble(value)

61. 将当前字符串转换为Double数值，如果转换失败，则返回指定值，例如指定值配置为1.1d。

    表达式：NumberUtils.toDouble(value,*1.1d*)

62. 将当前字符串转换为Float数值，如果转换失败，则返回0.0f。

    表达式：NumberUtils.toFloat(value)

63. 将当前字符串转换为Float数值，如果转换失败，则返回指定值，例如配置指定值为1.1f。

    表达式：NumberUtils.toFloat(value,*1.1f*)

64. 将当前字符串转换为Int数值，如果转换失败，则返回0。

    表达式：NumberUtils.toInt(value)

65. 将当前字符串转换为Int数值，如果转换失败，则返回指定值，例如配置指定值为1。

    表达式：NumberUtils.toInt(value,*1*)

66. 将字符串转换为Long数值，如果转换失败，则返回0。

    表达式：NumberUtils.toLong(value)

67. 将当前字符串转换为Long数值，如果转换失败，则返回指定值，例如配置指定值为1L。

    表达式：NumberUtils.toLong(value,*1L*)

68. 将字符串转换为Short数值，如果转换失败，则返回0。

    表达式：NumberUtils.toShort(value)

69. 将当前字符串转换为Short数值，如果转换失败，则返回指定值，例如配置指定值为1。

    表达式：NumberUtils.toShort(value,*1*)

70. 将当前IP字符串转换为Long数值，例如将“10.78.124.0”转换为LONG数值是“172915712”。

    表达式：CommonUtils.ipToLong(value)

71. 从网络读取一个IP与物理地址映射文件，并存放到Map集合，这里的URL是IP与地址映射文件存放地址，例如“http://10.114.205.45:21203/sqoop/IpList.csv”。

    表达式：HttpsUtils.downloadMap("*url*")

72. 将IP与地址映射对象缓存起来并指定一个key值用于检索，例如“ipList”。

    表达式：CommonUtils.setCache("*ipList*",HttpsUtils.downloadMap("*url*"))

73. 取出缓存的IP与地址映射对象。

    表达式：CommonUtils.getCache("*ipList*")

74. 判断是否有IP与地址映射缓存。

    表达式：CommonUtils.cacheExists("*ipList*")

75. 根据IP取出对应的详细地址：国家_省份_城市_运营商，例如“1xx.78.124.0”对应的地址为“中国_广东_深圳_电信”，取不到对应地址则默认“**_**_**_**”。如果需要，可通过StringUtil类表达式对地址进行进一步拆分。

    表达式：
    CommonUtils.getMapValue(CommonUtils.ipToLong(value),CommonUtils.cacheExists("*ipLis*")?
    CommonUtils.getCache("*ipLis*"):CommonUtils.setCache("*ipLis*",HttpsUtils.downloadMap("*url*")))

76. 根据指定的偏移类型（month/day/hour/minute/second）及偏移量（正数表示增加，负数表示减少），将指定格式的时间转换为一个新时间，例如将“2019-05-21 12:00:00”增加8个小时。

    表达式：DateUtils.getCurrentTimeByZone("*yyyy-MM-dd HH:mm:ss*",value, "*hour*", *8*)

2.3 数据源为 Hive 时支持哪些数据格式?

云数据迁移服务支持从Hive数据源读写的数据格式包括SequenceFile、TextFile、ORC、Parquet。

2.4 是否支持同步作业到其他集群?

CDM服务虽然不支持直接在不同集群间迁移作业，但是通过批量导出、批量导入作业的功能，可以间接实现集群间的作业迁移，方法如下：

1. 将CDM集群1中的所有作业批量导出，将作业的JSON文件保存到本地。

   由于安全原因，CDM导出作业时没有导出连接密码，连接密码全部使用“Add password here”替换。
2. 在本地编辑JSON文件，将“Add password here”替换为对应连接的正确密码。
3. 将编辑好的JSON文件批量导入到CDM集群2，实现集群1和集群2之间的作业同步。

2.5 是否支持批量创建作业?

CDM服务可以通过批量导入的功能，实现批量创建作业，方法如下：

1. 手动创建一个作业。
2. 导出作业，将作业的JSON文件保存到本地。
3. 编辑JSON文件，参考该作业的配置，在JSON文件中批量复制出更多作业。
4. 将JSON文件导入CDM集群，实现批量创建作业。

2.6 是否支持批量调度作业?

问题描述

cdm 下作业很多，是否支持统一调度所有的作业?

解决方法

1. 访问智能数据湖运营平台的数据开发模块。
2. 在数据开发主界面的左侧导航栏，选择“数据开发 > 作业开发”，新建作业。
3. 拖动多个CDM Job节点至画布，然后再编排作业。

2.7 如何将作业备份?

可以，如果用户长时间不需要使用CDM集群，可以将CDM集群停掉或删除来降低成本。

删除前，用户可以先通过CDM的批量导出功能，把所有作业脚本保存到本地，仅在需要的时候再重新创建集群、重新导入作业，实现作业备份。

2.8 如何使用 Java 调用 CDM 的 Rest API 创建数据迁移作业?

CDM提供了Rest API，可以通过程序调用实现自动化的作业创建或执行控制。

这里以CDM迁移MySQL数据库的表city1的数据到DWS的表city2为例，介绍如何使用Java调用CDM服务的REST API创建、启动、查询、删除该CDM作业。

需要提前准备以下数据：

1. 云账号的用户名、账号名和项目ID。
   获取方法：登录**“我的凭证”**界面，可获取用户名、账号名，在项目列表中可获取对应区域的“项目ID”，例如“1af30ca47b5a4eb987e325a846458b7a”。
2. 创建一个CDM集群，并获取集群ID。
   获取方法：在集群管理界面，单击CDM集群名称可查看集群ID，例如“c110beff-0f11-4e75-8b10-da7cd882b0ef”。
3. 创建一个MySQL数据库和一个DWS数据库，并创建好表city1和表city2，创表语句如下：
   ```
   MySQL:
   create table city1(code varchar(10),name varchar(32));
   insert into city1 values('NY','New York');
   DWS:
   create table city2(code varchar(10),name varchar(32));
   ```
4. 在CDM集群下，创建连接到MySQL的连接，例如连接名称为“mysqltestlink”。创建连接到DWS的连接，例如连接名称为“dwstestlink”。
5. 运行下述代码，依赖HttpClient包，建议使用4.5版本。Maven配置如下：
   ```
   <project>
   <modelVersion>4.0.0</modelVersion>
   <groupId>cdm</groupId>
   <artifactId>cdm-client</artifactId>
   <version>1</version>
   <dependencies>
   <dependency>
   <groupId>org.apache.httpcomponents</groupId>
   <artifactId>httpclient</artifactId>
   <version>4.5</version>
   </dependency>
   </dependencies>
   </project>
   ```

代码示例

使用Java调用CDM服务的REST API创建、启动、查询、删除CDM作业的代码示例如下：


```
package cdmclient;
import java.io.IOException;
import org.apache.http.Header;
import org.apache.http.HttpEntity;
import org.apache.http.HttpHost;
import org.apache.http.auth.AuthScope;
import org.apache.http.auth.UsernamePasswordCredentials;
import org.apache.http.client.CredentialsProvider;
import org.apache.http.client.config.RequestConfig;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpDelete;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.client.methods.HttpPut;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.BasicCredentialsProvider;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
public class CdmClient {
private final static String DOMAIN_NAME="云账号名";
private final static String USER_NAME="云用户名";
private final static String USER_PASSWORD="云用户密码";
private final static String PROJECT_ID="项目ID";
private final static String CLUSTER_ID="CDM集群ID";
private final static String JOB_NAME="作业名称";
private final static String FROM_LINKNAME="源连接名称";
private final static String TO_LINKNAME="目的连接名称";
private final static String IAM_ENDPOINT="IAM的Endpoint";
private final static String CDM_ENDPOINT="CDM的Endpoint";
private CloseableHttpClient httpclient;
private String token;

public CdmClient() {
this.httpclient = createHttpClient();
this.token = login();
}

private CloseableHttpClient createHttpClient() {
CloseableHttpClient httpclient =HttpClients.createDefault();
return httpclient;
}

private String login(){
HttpPost httpPost = new HttpPost("https://"+IAM_ENDPOINT+"/v3/auth/tokens");
String json =
"{\r\n"+
"\"auth\": {\r\n"+
"\"identity\": {\r\n"+
"\"methods\": [\"password\"],\r\n"+
"\"password\": {\r\n"+
"\"user\": {\r\n"+
"\"name\": \""+USER_NAME+"\",\r\n"+
"\"password\": \""+USER_PASSWORD+"\",\r\n"+
"\"domain\": {\r\n"+
"\"name\": \""+DOMAIN_NAME+"\"\r\n"+
"}\r\n"+
"}\r\n"+
"}\r\n"+
"},\r\n"+
"\"scope\": {\r\n"+
"\"project\": {\r\n"+
```

```
"\"name\": \"PROJECT_NAME\"\r\n"+
"}\r\n"+
"}\r\n"+
"}\r\n"+
"}\r\n";
try {
StringEntity s = new StringEntity(json);
s.setContentEncoding("UTF-8");
s.setContentType("application/json");
httpPost.setEntity(s);
CloseableHttpResponse response = httpclient.execute(httpPost);
Header tokenHeader = response.getFirstHeader("X-Subject-Token");
String token = tokenHeader.getValue();
System.out.println("Login successful");
return token;
} catch (Exception e) {
throw new RuntimeException("login failed.", e);
}
}
/*创建作业*/

public void createJob(){
HttpPost httpPost = new HttpPost("https://"+CDM_ENDPOINT+"/cdm/v1.0/"+PROJECT_ID+"/
clusters/"+CLUSTER_ID+"/cdm/job");

/**此处JSON信息比较复杂，可以先在作业管理界面上创建一个作业，然后单击作业后的“作业JSON
定义”，复制其中的JSON内容，格式化为Java字符串语法，然后粘贴到此处。
*JSON消息体中一般只需要替换连接名、导入和导出的表名、导入导出表的字段列表、源表中用于分
区的字段。**/

String json =
"{\r\n"+
"\"jobs\": [\r\n"+
"{\r\n"+
"\"from-connector-name\": \"generic-jdbc-connector\",\r\n"+
"\"name\": \""+JOB_NAME+"\",\r\n"+
"\"to-connector-name\": \"generic-jdbc-connector\",\r\n"+
"\"driver-config-values\": {\r\n"+
"\"configs\": [\r\n"+
"{\r\n"+
"\"inputs\": [\r\n"+
"{\r\n"+
"\"name\": \"throttlingConfig.numExtractors\",\r\n"+
"\"value\": \"1\"\r\n"+
"}\r\n"+
"],\r\n"+
"\"validators\": [],\r\n"+
"\"type\": \"JOB\",\r\n"+
"\"id\": 30,\r\n"+
"\"name\": \"throttlingConfig\"\r\n"+
"}\r\n"+
"]\r\n"+
"},\r\n"+
"\"from-link-name\": \""+FROM_LINKNAME+"\",\r\n"+
"\"from-config-values\": {\r\n"+
"\"configs\": [\r\n"+
"{\r\n"+
"\"inputs\": [\r\n"+
"{\r\n"+
"\"name\": \"fromJobConfig.schemaName\",\r\n"+
"\"value\": \"sqoop\"\r\n"+
"},\r\n"+
```

```
"{\r\n"+
"\"name\": \"fromJobConfig.tableName\",\r\n"+
"\"value\": \"city1\"\r\n"+
"},\r\n"+
"{\r\n"+
"\"name\": \"fromJobConfig.columnList\",\r\n"+
"\"value\": \"code&name\"\r\n"+
"},\r\n"+
"{\r\n"+
"\"name\": \"fromJobConfig.partitionColumn\",\r\n"+
"\"value\": \"code\"\r\n"+
"}\r\n"+
"],\r\n"+
"\"validators\": [],\r\n"+
"\"type\": \"JOB\",\r\n"+
"\"id\": 7,\r\n"+
"\"name\": \"fromJobConfig\"\r\n"+
"}\r\n"+
"]\r\n"+
"},\r\n"+
"\"to-link-name\": \""+TO_LINKNAME+"\",\r\n"+
"\"to-config-values\": {\r\n"+
"\"configs\": [\r\n"+
"{\r\n"+
"\"inputs\": [\r\n"+
"{\r\n"+
"\"name\": \"toJobConfig.schemaName\",\r\n"+
"\"value\": \"sqoop\"\r\n"+
"},\r\n"+
"{\r\n"+
"\"name\": \"toJobConfig.tableName\",\r\n"+
"\"value\": \"city2\"\r\n"+
"},\r\n"+
"{\r\n"+
"\"name\": \"toJobConfig.columnList\",\r\n"+
"\"value\": \"code&name\"\r\n"+
"}, \r\n"+
"{\r\n"+
"\"name\": \"toJobConfig.shouldClearTable\",\r\n"+
"\"value\": \"true\"\r\n"+
"}\r\n"+
"],\r\n"+
"\"validators\": [],\r\n"+
"\"type\": \"JOB\",\r\n"+
"\"id\": 9,\r\n"+
"\"name\": \"toJobConfig\"\r\n"+
"}\r\n"+
"]\r\n"+
"}\r\n"+
"}\r\n"+
"]\r\n"+
"}\r\n";
try {
StringEntity s = new StringEntity(json);
s.setContentEncoding("UTF-8");
s.setContentType("application/json");
httpPost.setEntity(s);
httpPost.addHeader("X-Auth-Token", this.token);
httpPost.addHeader("X-Language", "en-us");
CloseableHttpResponse response = httpclient.execute(httpPost);
int status = response.getStatusLine().getStatusCode();
if(status == 200){
```



```
System.out.println("Create job successful.");
}else{
System.out.println("Create job failed.");
HttpEntity entity = response.getEntity();
System.out.println(EntityUtils.toString(entity));
}
} catch (Exception e) {
e.printStackTrace();
throw new RuntimeException("Create job failed.", e);
}
}
/*启动作业*/

public void startJob(){
HttpPut httpPut = new HttpPut("https://"+CDM_ENDPOINT+"/cdm/v1.0/"+PROJECT_ID+"/
clusters/"+CLUSTER_ID+"/cdm/job/"+JOB_NAME+"/start");
String json = "";
try {
StringEntity s = new StringEntity(json);
s.setContentEncoding("UTF-8");
s.setContentType("application/json");
httpPut.setEntity(s);
httpPut.addHeader("X-Auth-Token", this.token);
httpPut.addHeader("X-Language", "en-us");
CloseableHttpResponse response = httpclient.execute(httpPut);
int status = response.getStatusLine().getStatusCode();
if(status == 200){
System.out.println("Start job successful.");
}else{
System.out.println("Start job failed.");
HttpEntity entity = response.getEntity();
System.out.println(EntityUtils.toString(entity));
}
} catch (Exception e) {
e.printStackTrace();
throw new RuntimeException("Start job failed.", e);
}
}
/*循环查询作业运行状态，直到作业运行结束。*/

public void getJobStatus(){
HttpGet httpGet = new HttpGet("https://"+CDM_ENDPOINT+"/cdm/v1.0/"+PROJECT_ID+"/
clusters/"+CLUSTER_ID+"/cdm/job/"+JOB_NAME+"/status");
try {
httpGet.addHeader("X-Auth-Token", this.token);
httpGet.addHeader("X-Language", "en-us");
boolean flag = true;
while(flag){
CloseableHttpResponse response = httpclient.execute(httpGet);
int status = response.getStatusLine().getStatusCode();
if(status == 200){
HttpEntity entity = response.getEntity();
String msg = EntityUtils.toString(entity);
if(msg.contains("\"status\":\"SUCCEEDED\"")){
System.out.println("Job succeeded");
break;
}else if (msg.contains("\"status\":\"FAILED\"")){
System.out.println("Job failed.");
break;
}else{
Thread.sleep(1000);
}
```

```
}else{
System.out.println("Get job status failed.");
HttpEntity entity = response.getEntity();
System.out.println(EntityUtils.toString(entity));
break;
}
}
} catch (Exception e) {
e.printStackTrace();
throw new RuntimeException("Get job status failed.", e);
}
}
/*删除作业*/

public void deleteJob(){
HttpDelete httpDelte = new HttpDelete("https://"+CDM_ENDPOINT+"/cdm/v1.0/"+PROJECT_ID
+"/clusters/"+CLUSTER_ID+"/cdm/job/"+JOB_NAME);
try {
httpDelte.addHeader("X-Auth-Token", this.token);
httpDelte.addHeader("X-Language", "en-us");
CloseableHttpResponse response = httpclient.execute(httpDelte);
int status = response.getStatusLine().getStatusCode();
if(status == 200){
System.out.println("Delete job successful.");
}else{
System.out.println("Delete job failed.");
HttpEntity entity = response.getEntity();
System.out.println(EntityUtils.toString(entity));
}
} catch (Exception e) {
e.printStackTrace();
throw new RuntimeException("Delete job failed.", e);
}
}
/*关闭*/

public void close(){
try {
httpclient.close();
} catch (IOException e) {
throw new RuntimeException("Close failed.", e);
}
}

public static void main(String[] args){
CdmClient cdmClient = new CdmClient();
cdmClient.createJob();
cdmClient.startJob();
cdmClient.getJobStatus();
cdmClient.deleteJob();
cdmClient.close();
}
}
```

2.9 如何将云下内网或第三方云上的私网与 CDM 连通?

很多企业会把关键数据源建设在内网，例如数据库、文件服务器等。由于CDM服务运行在云上，如果要通过CDM服务迁移内网数据到云上的话，可以通过以下几种方式连通内网和CDM的网络：

1. 为内网数据源节点绑定Internet IP地址，允许CDM通过Internet直接访问。
2. 在本地数据中心和云服务VPC之间建立VPN通道。
3. 通过专线连接数据中心和云服务。
4. 通过NAT（网络地址转换，Network Address Translation）或端口转发，以代理的方式访问。

这里重点介绍如何通过端口转发工具来实现访问内部数据，流程如下：

1. 找一台windows机器作为网关，该机器必须可以直接访问Internet，同时可以访问内网。
2. 在该机器上安装端口映射工具（IPOP）。
3. 通过端口映射工具（IPOP）配置端口映射。

须知

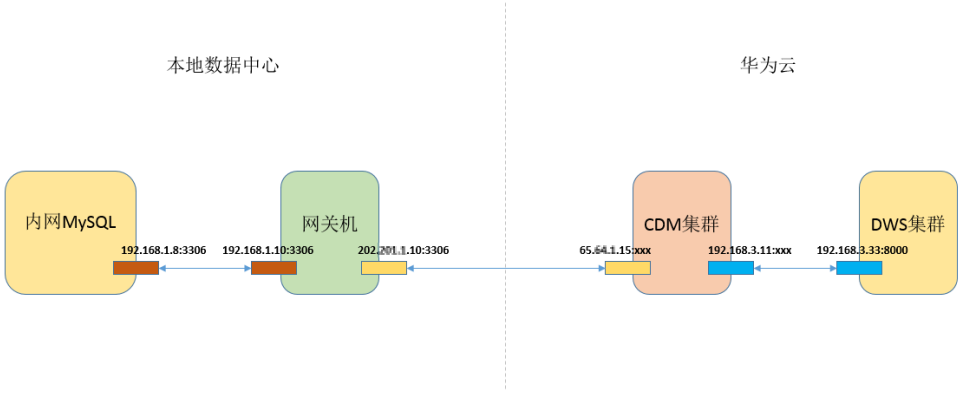
长时间将内网数据库暴露在公网会有安全风险，迁移数据完成后，请及时停止端口映射。

场景描述

这里假设是将内网MySQL迁移到云服务DWS，网络拓扑样例如图2-3所示。

图中的内网既可以是企业自己的数据中心，也可以是在第三方云的虚拟数据中心私网。

图 2-3 网络拓扑样例



操作步骤

步骤1 找一台Windows机器作为网关机，该机器同时配置内网和外网IP。通过以下测试来确保网关机器的服务要求：

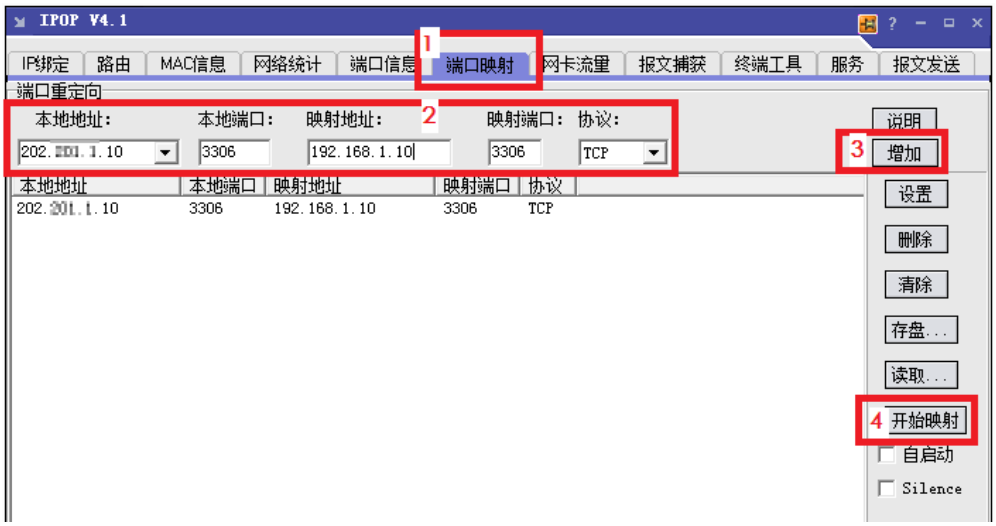
1. 在该机器上**ping**内网MySQL地址可以**ping**通，例如：**ping 192.168.1.8**。
2. 在另外一台可上网的机器上**ping**网关机的公网地址可以**ping**通，例如**ping 202.*xx.xx*.10**。

步骤2 下载端口映射工具IPOP，在网关机上安装IPOP。

步骤3 运行端口映射工具，选择“端口映射”，如图2-4所示。

- 本地地址、本地端口：配置为网关机的公网地址和端口（后续在CDM上创建MySQL连接时输入这个地址和端口）。
- 映射地址、映射端口：配置为内网MySQL的地址和端口。

图 2-4 配置端口映射



步骤4 单击“增加”，添加端口映射关系。

步骤5 单击“开始映射”，这时才会真正开始映射，接收数据包。

至此，就可以在CDM上通过弹性IP读取本地内网MySQL的数据，然后导入到云服务DWS中。

说明

1. CDM要访问本地数据源，也必须给CDM集群配置EIP。
2. 一般云服务DWS默认也是只允许VPC内部访问，创建CDM集群时，必须将CDM的VPC与DWS配置一致，且推荐在同一个内网和安全组，如果不同，还需要配置允许两个安全组之间的数据访问。
3. 端口映射不仅可以用于迁移内网数据库的数据，还可以迁移例如SFTP服务器上的数据。
4. Linux机器也可以通过IPTABLE实现端口映射。
5. 内网中的FTP通过端口映射到公网时，需要检查是否启用了PASV模式。这种情况下客户端和服务端建立连接的时候是走的随机端口，所以除了配置21端口映射外，还需要配置PASV模式的端口范围映射，例如vsftp通过配置pasv_min_port和pasv_max_port指定端口范围。

----结束

2.10 同一 VPC 和不同 VPC 的迁移性能?

传输速率取决带宽和文件读写速度。

3 故障处理类

3.1 日志提示解析日期格式失败时怎么处理?

问题描述

在使用CDM迁移其他数据源到云搜索服务（Cloud Search Service）的时候，作业执行失败，日志提示“Unparseable date”，如图3-1所示。

图 3-1 日志提示信息

```
java.text.ParseException: Unparseable date: "2018/01/05 15:15:46"
        at java.text.DateFormat.parse(DateFormat.java:366) ~[na:1.8.0_112]
        at org.apache.sqoop.connector.common.DataTypeUtil.convertDateFormat
        at org.apache.sqoop.connector.elasticsearch.ElasticSearchLoader.toJ
        at org.apache.sqoop.connector.elasticsearch.ElasticSearchLoader.arr
7]
        at org.apache.sqoop.connector.elasticsearch.ElasticSearchLoader.loa
```

原因分析

云搜索服务对于时间类型有一个特殊处理：如果存储的时间数据不带时区信息，在Kibana可视化的时候，Kibana会认为该时间为GMT标准时间。

在中国区则会产生显示时间少8小时的现象。因此在CDM迁移数据到云搜索服务的时候，如果是通过CDM自动创建的索引和类型（例如图3-2中，目的端的“date_test”和“test1”在云搜索服务中不存在时，CDM会在云搜索服务中自动创建该索引和类型），则CDM默认会将时间类型字段的格式设置为“yyyy-MM-dd HH:mm:ss.SSS Z”的标准格式，例如“2018-01-08 08:08:08.666 +0800”。

图 3-2 作业配置

此时，从其他数据源导入数据到云搜索服务时，如果源端数据中的日期格式不完全满足标准格式，例如“2018/01/05 15:15:46”，则CDM作业会执行失败，日志提示无法解析日期格式。需要通过CDM配置字段转换器，将日期字段的格式转换为云搜索服务的目的端格式。

解决方法

1. 编辑作业，进入作业的字段映射步骤，在源端的时间格式字段后面，选择新建转换器，如图3-3所示。

   图 3-3 新建转换器

2. 转换器类型选择“表达式转换”，目前表达式转换支持字符串和日期类型的函数，语法和Java的字符串和时间格式函数非常相似，可以查看表达式转换了解如何编写表达式。
3. 本例中源时间格式是“yyyy/MM/dd HH:mm:ss”，要将其转换成“yyyy-MM-dd HH:mm:ss.SSS Z”，需要经过如下几步：
   a. 添加时区信息“+0800”到原始日期字符串的尾部，对应的表达式为：**value +" +0800"**。

b. 使用原始日期格式来解析字符串，将字符串解析为一个日期对象。可以使用DateUtils.parseDate函数来解析，语法是：**DateUtils.parseDate(String value, String format)**。

c. 将日期对象格式化成目标格式的字符串，可以使用DateUtils.format函数来格式化，语法是**DateUtils.format(Date date, String format)**。

因此本例中串起来完整的表达式是：
DateUtils.format(DateUtils.parseDate(value+" +0800","yyyy/MM/dd HH:mm:ss Z"),"yyyy-MM-dd HH:mm:ss.SSS Z")，如图3-4所示。

图 3-4 配置表达式

4. 保存转换器配置，再保存并运行作业，可解决云搜索服务的解析日期格式失败问题。

3.2 字段映射界面无法显示所有列怎么处理?

问题描述

在使用CDM从HBase/CloudTable导出数据时，在字段映射界面HBase/CloudTable表的字段偶尔显示不全，无法与目的端字段一一匹配，造成导入到目的端的数据不完整。

原因分析

由于HBase/CloudTable无Schema，每条数据的列数不固定，在字段映射界面CDM通过获取样值的方式有较大概率无法获得所有列，此时作业执行完后会造成目的端的数据不全。

这个问题，可以通过以下方法解决：

1. 在CDM的字段映射界面增加字段。
2. 在CDM的作业管理界面直接编辑作业的JSON（修改“fromJobConfig.columns”、“toJobConfig.columnList”这2个参数）。

3. 导出作业的JSON文件到本地，在本地手动修改JSON文件中的参数后（原理同2相同），再导回CDM。

推荐使用方法1，下面以HBase导到DWS为例进行说明。

解决方法一：CDM 的字段映射界面增加字段

1. 获取源端HBase待迁移的表中所有的字段，列族与列之间用“:”分隔，例如：

```
rowkey:rowkey
g:DAY_COUNT
g:CATEGORY_ID
g:CATEGORY_NAME
g:FIND_TIME
g:UPLOAD_PEOPLE
g:ID
g:INFOMATION_ID
g:TITLE
g:COORDINATE_X
g:COORDINATE_Y
g:COORDINATE_Z
g:CONTENT
g:IMAGES
g:STATE
```

2. 在CDM的作业管理界面，找到HBase导出数据到DWS的作业，单击作业后面的“编辑”，进入字段映射界面，如图3-5所示。

图 3-5 字段映射 03

3. 单击添加字段，在弹出框中选择“添加新字段”，如图3-6所示。

图 3-6 添加字段 04

说明

- 添加完字段后，新增的字段在界面不显示样值，这个不影响字段值的传输，CDM会将字段值直接写入目的端。
- 这里“添加新字段”的功能，要求源端数据源为：MongoDB、HBase、关系型数据库或Redis，其中Redis必须为Hash数据格式。

4. 全部字段添加完之后，检查源端和目的端的字段映射关系是否正确，如果不正确可以拖拽字段调整字段位置。
5. 单击“下一步”后保存作业。

解决方法二：修改 JSON 文件

1. 获取源端HBase待迁移的表中所有的字段，列族与列之间用“:”分隔，例如：

```
rowkey:rowkey
g:DAY_COUNT
g:CATEGORY_ID
g:CATEGORY_NAME
g:FIND_TIME
g:UPLOAD_PEOPLE
g:ID
g:INFOMATION_ID
g:TITLE
g:COORDINATE_X
g:COORDINATE_Y
g:COORDINATE_Z
g:CONTENT
g:IMAGES
g:STATE
```

2. 在DWS目的表中，获取与HBase表对应的字段。

   如果DWS目的表中没有HBase对应的字段名，需在DWS表定义中加上，假设DWS表中的字段齐全且如下：

```
rowkey
day_count
category
category_name
find_time
upload_people
```



```
id
infomation_id
title
coordinate_x
coordinate_y
coordinate_z
content
images
state
```

3. 在CDM的作业管理界面，找到HBase到DWS的作业，选择作业后面的“更多 > 编辑作业JSON”。
4. 在CDM界面编辑作业的JSON文件。
   a. 修改源端的“fromJobConfig.columns”参数，配置为**1**获取的HBase的字段，列号之间使用“&”分隔，列族与列之间用“:”分隔，如下：

```
"from-config-values": {
            "configs": [
               {
                  "inputs": [
                     {
                        "name": "fromJobConfig.table",
                        "value": "HBase"
                     },
                     {
                        "name": "fromJobConfig.columns",
                        "value":
"rowkey:rowkey&g:DAY_COUNT&g:CATEGORY_ID&g:CATEGORY_NAME&g:FIND_TIME&g:UPLOAD_PEOPLE&g:ID&g:INFOMATION_ID&g:TITLE&g:COORDINATE_X&g:COORDINATE_Y&g:COORDINATE_Z&g:CONTENT&g:IMAGES&g:STATE"
                     },
                     {
                        "name": "fromJobConfig.formats",
                        "value": {
                           "2": "yyyy-MM-dd",
                           "undefined": "yyyy-MM-dd"
                        }
                     }
                  ],
                  "name": "fromJobConfig"
               }
            ]
         }
```

   b. 修改目的端的“toJobConfig.columnList”参数，配置为**2**中DWS的字段列表。

      这里的顺序必须与HBase保持一致，才能保证正确的字段映射关系，字段名之间使用“&”分隔，如下：

```
"to-config-values": {
            "configs": [
               {
                  "inputs": [
                     {
                        "name": "toJobConfig.schemaName",
                        "value": "dbadmin"
                     },
                     {
                        "name": "toJobConfig.tablePreparation",
                        "value": "DO_NOTHING"
                     },
                     {
                        "name": "toJobConfig.tableName",
                        "value": "DWS "
                     },
                     {
                        "name": "toJobConfig.columnList",
                        "value":
"rowkey&day_count&category&category_name&find_time&upload_people&id&infomation_
```



```
id&title&coordinate_x&coordinate_y&coordinate_z&content&images&state"
                    },
                    {
                        "name": "toJobConfig.shouldClearTable",
                        "value": "true"
                    }
                ],
                "name": "toJobConfig"
            }
        ]
    }
```

c. 其他参数保持不变，单击“保存并运行”。

5. 作业完成后，查询DWS表中的数据是否和HBase中的数据匹配。如果不匹配，请检查JSON文件中HBase和DWS字段的顺序是否一致。

3.3 CDM 迁移数据到 DWS 时如何选取分布列?

在使用CDM迁移数据到数据仓库服务（DWS）或者FusionInsight LibrA，且CDM在DWS端自动创建一个新表时，在创建作业的字段映射界面，需要选择分布列，如图3-7所示。

图 3-7 选取分布列

由于分布列的选取，对于DWS/FusionInsight LibrA的运行非常重要，在CDM数据迁移到DWS/FusionInsight LibrA过程中，建议按如下顺序选取分布列：

1. 有主键可以使用主键作为分布列。
2. 多个数据段联合做主键的场景，建议设置所有主键作为分布列。
3. 在没有主键的场景下，如果没有选择分布列，DWS会默认第一列作为分布列，可能会有数据倾斜风险。

因此，在单表或整库导入到DWS/FusionInsight LibrA时，建议您在此处手动选择分布列，如果您没有选择，CDM会自动选择一个分布列。关于分布列的更多信息，请参见**数据仓库服务**。

DWS主键或表只有一个字段时，要求字段类型必须是如下常用的字符串、数值、日期类型。从其他数据库迁移到DWS时，如果选择自动建表，主键必须为以下类型，未设置主键的情况下至少要有一个字段是以下类型，否则会无法创建表导致CDM作业失败。

- INTEGER TYPES：TINYINT，SMALLINT，INT，BIGINT，NUMERIC/DECIMAL
- CHARACTER TYPES：CHAR，BPCHAR，VARCHAR，VARCHAR2，NVARCHAR2，TEXT

- DATA/TIME TYPES：DATE，TIME，TIMETZ，TIMESTAMP，TIMESTAMPTZ，INTERVAL，SMALLDATETIME

3.4 迁移到 DWS 时出现 value too long for type character varying 怎么处理?

问题描述

在使用CDM迁移数据到数据仓库服务（DWS）或者FusionInsight LibrA时，如果迁移作业失败，且执行日志中出现“value too long for type character varying”错误提示，如图3-8所示。

图 3-8 日志信息

```
Caused by: org.postgresql.util.PSQLException: ERROR: value too long for type character varying(50)
  Where: COPY fl_behavior_module, line 72, column MODULE_NAME: "[redacted]"
        at org.postgresql.core.v3.QueryExecutorImpl.receiveErrorResponse(QueryExecutorImpl.java:2477)
        at org.postgresql.core.v3.QueryExecutorImpl.processCopyResults(QueryExecutorImpl.java:1107)
        at org.postgresql.core.v3.QueryExecutorImpl.writeToCopy(QueryExecutorImpl.java:989)
        at org.postgresql.core.v3.CopyInImpl.writeToCopy(CopyInImpl.java:35)
        ... 16 common frames omitted
```

原因分析

这种情况一般是在迁移到DWS时数据有中文，且创建作业时选择了目的端自动建表的情况下。原因是DWS的varchar类型是按字节计算长度，一个中文字符在UTF-8编码下可能要占3个字节。当中文字符的字节超过DWS的varchar的长度时，就会出现错误：value too long for type character varying。

解决方法

这个问题，可以通过将目的端作业参数“扩大字符字段长度”选择“是”来解决，选择此选项后，再创建目的表时会自动将varchar类型的字段长度扩大3倍。

编辑CDM的表/文件迁移作业，目的端作业配置下“自动创表”选择“不存在时创建”，则高级属性下面会出现参数“扩大字符字段长度”，配置该参数为“是”即可，如图3-9所示。

图 3-9 扩大字符字段长度

3.5 OBS 导入数据到 SQL Server 时出现 Unable to execute the SQL statement 怎么处理?

问题描述

使用CDM从OBS导入数据到SQL Server时，作业运行失败，错误提示为：Unable to execute the SQL statement. Cause : 将截断字符串或二进制数据。

原因分析

用户OBS中的数据超出了SQL Server数据库的字段长度限制。

解决方法

在SQL Server数据库中建表时，将数据库字段改大，长度不能小于源端OBS中的数据长度。