云享家 | “别了XX”，不能别了你的数据库！

宋彬涛 BespinGlobal 1周前

在全民业务上云的今天，数据库这个大后方作为业务和应用支撑的弹药库，重要性就不言而喻了。然而数据库迁移的正确姿势，你真的Get了么？yugong（意译：愚公）项目是阿里的开源项目，该项目使用纯Java开发，主要作用是进行数据库迁移，目前该项目主要支持从oracle数据库向Mysql和DRDS数据库进行迁移。本期的“云享家”将为您解析Oracle在线迁移云平台的具体技术细节。

环境要求

操作系统

a. 纯java开发，有bat和shell脚本，windows/linux均可支持.
b. jdk建议使用1.6.25以上的版本，稳定可靠，目前阿里巴巴使用基本为此版本。

数据库

a. 源库为Oracle，目标库可为mysql/drds/Oracle. 基于标准jdbc协议开发，对数据库暂无版本要求。

需要的数据库账户权限:

1. 源库(oracle)

GRANT SELECT,INSERT,UPDATE,DELETE ON XXX TO XXX; #常见CRUD权限

GRANT CREATE ANY MATERIALIZED VIEW TO XXX;

GRANT DROP ANY MATERIALIZED VIEW TO XXX;

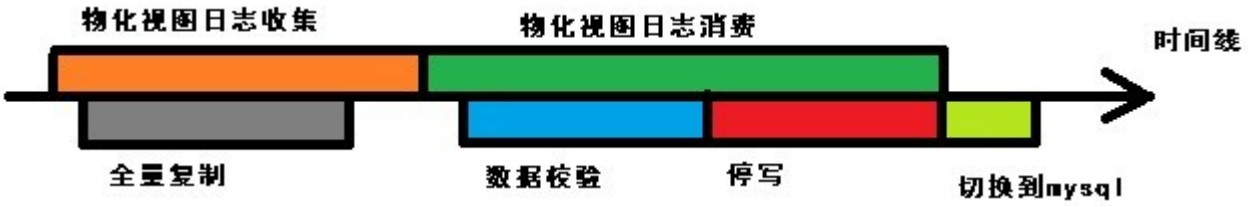
2. 目标库(mysql/oracle)

GRANT SELECT,INSERT,UPDATE,DELETE ON XXX TO XXX;

项目介绍

整个迁移方案，分为两部分：

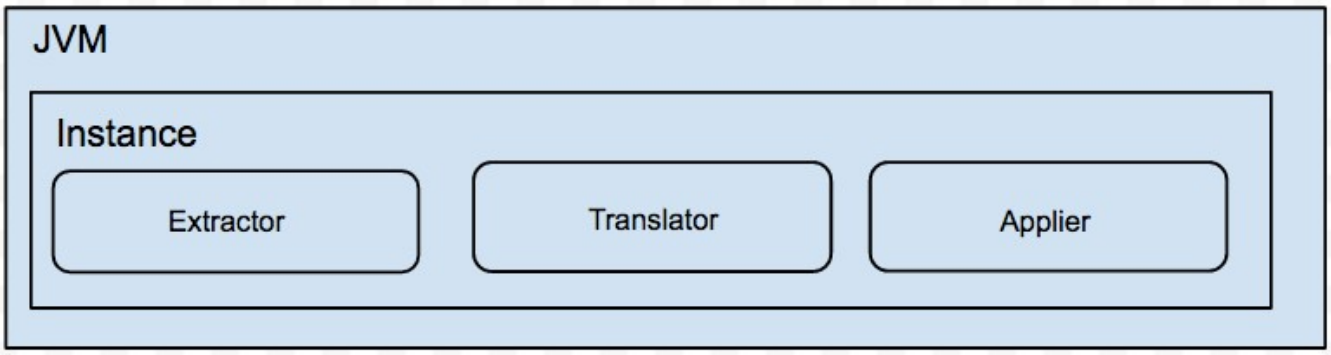
1. 全量迁移
2. 增量迁移



过程描述：

1. 增量数据收集 (创建oracle表的增量物化视图)
2. 进行全量复制
3. 进行增量复制 (并行进行数据校验)
4. 原库停写，切到新库

架构



1.一个Jvm Container对应多个instance，每个instance对应于一张表的迁移任务

2.instance分为三部分

- extractor（从源数据库上提取数据，可分为全量/增量实现)
- translator（将源库上的数据按照目标库的需求进行自定义转化)
- applier（将数据更新到目标库，可分为全量/增量/对比的实现)

部署

下载

github地址: https://github.com/alibaba/yugong

```
git clone https://github.com/alibaba/yugong.git
```

目录结构

解压缩发布包后，可得如下目录结构：

drwxr-xr-x 2 root root 4096 Jun 29 15:32 bin

drwxr-xr-x 4 root root 4096 Jun 29 15:23 conf

drwxr-xr-x 2 root root 4096 Jun 19 15:06 lib

-rw-r--r-- 1 root root 18045 Mar 2 2016 LICENSE

drwxrwxrwx 4 root root 4096 Jun 29 15:10 logs

-rw-r--r-- 1 root root 3767 Mar 5 2016 README.md

配置文件说明

默认值

参数名字	参数说明	默认值
数据库配置相关		
yugong.database.source.username yugong.database.source.password yugong.database.source.type yugong.database.source.url yugong.database.source.encode	源数据库的相关账户和链接信息 driver url 示例： 1. ORACLE ：jdbc:oracle:thin:@10.20.144.29:1521:ointest 2. MYSQL ：jdbc:mysql://10.20.144.34:3306/test	encode默认为UTF-8，其他无默认值
yugong.database.target.username yugong.database.target.password yugong.database.target.type yugong.database.target.url yugong.database.target.	目标数据库的相关账户和链接信息	encode默认为UTF-8，其他无默认值

encode		
yugong.table.white	需要同步表，白名单，定义需要同步的表 几点说明： 1. 表名支持like匹配,比如'%'匹配一个或者多个字符,下划线'_'匹配单个字符,可以通过单斜杠\进行转义符定义. 2. 表明为schema+table name组成，多个表可加逗号分隔 3. 如果白名单为空，代表整个库所有表，否则按指定的表进行同步 例子: 1. yugong_example_% (可以匹配yugong_example打头的字符串) 2. alibaba.yugong_example_test_ (可以匹配alibaba.yugong_example_test1 / alibaba.yugong_example_test2)	无
yugong.table.black	需要同步表，黑名单，需要忽略同步的表 配置方式可参考yugong.table.white	无
yugong.table.mode	运行模式，目前支持的模式为： 1. MARK (开启增量记录，比如oracle就是创建物化视图) 2. FULL (全量模式) 3. INC (增量模式) 4. ALL (自动全量+增量模式) 5. CHECK (数据对比模式) 6. CLEAR (清理增量记录，比如ora	无

	cle就是删除物化视图)	
yugong.table.concurrent.enable	多张表之前是否开启并行处理，如果false代表需要串行处理	true
yugong.table.concurrent.size	允许并行处理的表数	5
yugong.table.retry.times	表同步出错后的重试次数	3
yugong.table.retry.interval	表同步出错后的重试时的时间间隔，单位ms	1000
yugong.table.batchApply	是否开启jdbc　batch处理	true
yugong.table.onceCrawNum	extractor/applier每个批次最多处理记录数	1000
yugong.table.tpslimit	tps限制，0代表不限制	0
yugong.table.ignoreSchema	是否忽略schema同步 (如果mysql和oracle对应的schema不同，可设置为true)	false
yugong.table.skipApplierException	true代表当applier出现数据库异常时，比如约束键冲突，可对单条出异常的数据进行忽略. 同时记录skiped　record data信息，日志中包含record的所有列信息，包括主键.	false
extractor配置相关		
yugong.extractor.dump	是否记录extractor提取到的所有数据	false
yugong.extractor.concurrent.enable	extractor是否开启并行处理，目前主要应用为增量模式反查源表	true
yugong.extractor.concurrent.global	extractor是启用全局线程池模式,如果true代表所有extractor任务都使用一组线程池,线程池大小由concurrent.size控制	false

查看日志

对应日志结构为：

```
logs/
 - yugong/  #系统根日志
    - table.log
 - ${table}/  #每张同步表的日志信息
    - table.log
    - extractor.log
    - applier.log
    - check.log
```

全量完成的日志：(会在yugong/table.log 和 ${table}/table.log中出现记录)

```
table[OTTER2.TEST_ALL_ONE_PK] is end!
```

增量日志：(会在${table}/table.log中出现记录)

```
table[OTTER2.TEST_ALL_ONE_PK] now is CATCH_UP ... #代表已经追上，最后一次增量数据小于onceCrawNum数量
table[OTTER2.TEST_ALL_ONE_PK] now is NO_UPDATE ... #代表最近一次无增量数据
```

ALL(全量+增量)模式日志： (会在${table}/table.log中出现记录)

```
table [OTTER2.TEST_ALL_ONE_PK] full extractor is end , next auto start inc extractor #出现这条代表全量已经完成，进入增量模式
```

CHECK日志： (会在${table}/check.log中出现diff记录)

```
-----------------
- Schema: yugong , Table: test_all_one_pk
-----------------
---Pks
        ColumnValue[column=ColumnMeta[index=0,name=ID,type=3],value=2576]
---diff
        ColumnMeta[index=3,name=AMOUNT,type=3] , values : [0] vs [0.0]
```

同步过程数据日志：会通过extractor.log/applier.log分别记录extractor和applier的数据记录，因为有DataTranslator的存在，两者记录可能不一致，所以分开两份记录。

统计信息：

- progress统计，会在主日志下，输出当前全量/增量/异常表的数据，可通过该日志，全局把握整个迁移任务的进度，输出类似：

{未启动:0,全量中:2,增量中:3,已追上:3,异常数:0}

- stat统计，会在每个表迁移日志下，输出当前迁移的tps信息

{总记录数:180000,采样记录数:5000,同步TPS:4681,最长时间:215,最小时间:212,平均时间:213}

切换流程

1. 当任务处于追上状态时候，表示已经处于实时同步状态
2. 后续通过源数据库进行停写，稍等1-2分钟后（保证延时的数据最终得到同步，此时源库和目标库当前数据是完全一致的）
3. 检查增量持续处于NO_UPDATE状态，可关闭该迁移任务(sh stop.sh)，即可发布新程序，使用新的数据库，完成切换的流程.

自定义数据转换

如果要迁移的oracle和mysql的表结构不同，比如表名，字段名有差异，字段类型不兼容，需要使用自定义数据转换。如果完全相同那就可以跳过此章节

整个数据流为：DB -> Extractor -> DataTranslator -> Applier -> DB，本程序预留DataTranslator接口，允许外部用户自定义数据处理逻辑，比如：

1. 表名不同
2. 字段名不同
3. 字段类型不同
4. 字段个数不同

运行过程join其他表的数据做计算等。

举例

```
/**
 * 一个迁移的例子，涵盖一些基本转换操作
 *
 * <pre>
 * 例子包含特性:
 *  1. schema/table 名不同. oracle 中为 otter2.yugong_example_oracle，mysql 中为 test.yugong_example_mysql
 * 2. 字段名字不同. oracle中的name字段，映射到mysql的display_name
 *  3. 字段逻辑处理. mysql 的 display_name 字段数据来源为 oracle 库的:name+'('alias_name+')'
 * 4. 字段类型不同. oracle中的amount为number类型，映射到mysql的amount为varchar文本型
 * 5. 源库多一个字段. oracle中多了一个alias_name字段
 * 6. 目标库多了一个字段. mysql中多了一个gmt_move字段，(简单的用迁移时的当前时间进行填充)
 *
 * 测试的表结构:
 * // oracle表
 * create table otter2.yugong_example_oracle
 * (
 *     id NUMBER(11) ,
 *     name varchar2(32) ,
 *     alias_name  char(32) default ' ' not null,
 *     amount number(11,2),
 *     score  number(20),
 *     text_b blob,
 *     text_c clob,
 *     gmt_create date not null,
 *     gmt_modified date not null,
 *     CONSTRAINT yugong_example_oracle_pk_id  PRIMARY  KEY (id)
 * );
 *
 * // mysql表
 * create table test.yugong_example_mysql
 * (
 *     id bigint(20) unsigned auto_increment,
```

```
*     display_name varchar(128) ,
*     amount varchar(32),
*     score bigint(20) unsigned ,
*     text_b blob,
*     text_c text,
*     gmt_create timestamp not null,
*     gmt_modified timestamp not null,
*     gmt_move timestamp not null,
*     CONSTRAINT yugong_example_mysql_pk_id  PRIMARY KEY (id)
* );
* </pre>
*
 */
public class YugongExampleOracleDataTranslator extends AbstractDataTranslator
implements DataTranslator {
public boolean translator(Record record) {
    // 1. schema/table名不同
    // record.setSchemaName("test");
    record.setTableName("yugong_example_mysql");

    if (record instanceof IncrementRecord) {
        if (IncrementOpType.D == ((IncrementRecord) record).getOpType()) {
            // 忽略delete
            return super.translator(record);
        }
    }

    // 2. 字段名字不同
    ColumnValue nameColumn = record.getColumnByName("name");
    nameColumn.getColumn().setName("display_name");

    // 3. 字段逻辑处理
    ColumnValue aliasNameColumn = record.getColumnByName("alias_name");
    StringBuilder displayNameValue = new StringBuilder(64);
    displayNameValue.append(ObjectUtils.toString(nameColumn.getValue()))
        .append('(')
        .append(ObjectUtils.toString(aliasNameColumn.getValue()))
        .append(')');
    nameColumn.setValue(displayNameValue.toString());

    // 4. 字段类型不同
```

```
    ColumnValue amountColumn = record.getColumnByName("amount");
    amountColumn.getColumn().setType(Types.VARCHAR);
    amountColumn.setValue(ObjectUtils.toString(amountColumn.getValue()));

    // 5. 源库多一个字段
    record.getColumns().remove(aliasNameColumn);

    // 6. 目标库多了一个字段
            ColumnMeta gmtMoveMeta = new ColumnMeta("gmt_move",
Types.TIMESTAMP);
    ColumnValue gmtMoveColumn = new ColumnValue(gmtMoveMeta, new Date());
    record.addColumn(gmtMoveColumn);

    // ColumnValue text_c = record.getColumnByName("text_c");
    // try {
    // text_c.setValue(new String((byte[]) text_c.getValue(), "GBK"));
    // } catch (UnsupportedEncodingException e) {
    // e.printStackTrace();
    // }
    return super.translator(record);
}

}
```

运行模式详细介绍

MARK模式(MARK)

开启增量日志的记录，如果是oracle就是创建物化视图。

CLEAR模式(CLEAR)

清理增量日志的记录，如果是oracle就是删除物化视图。

全量模式(FULL)

全量模式，顾名思议即为对源表进行一次全量操作，遍历源表所有的数据后，插入目标表。

全量有两种处理方式：

1. 分页处理：如果源表存在主键，只有一个主键字段，并且主键字段类型为Number类型，默认会选择该分页处理模式. 优点：支持断点续做，对源库压力相对较小。 缺点：迁移速度慢
2. once处理：通过select * from访问整个源表的某一个mvcc版本的数据，通过cursor.next遍历整个结果集. 优点：迁移速度快，为分页处理的5倍左右。 缺点：源库压力大，如果源库并发修改量大，会导致数据库MVCC版本过多，出现栈错误. 还有就是不支持断点续做.

特别注意

如果全量模式运行过程中，源库有变化时，不能保证源库最近变化的数据能同步到目标表，这时需要配合增量模式. 具体操作就是：在运行全量模式之前，先开启增量模式的记录日志功能，然后

开启全量模式，完成后，再将最近变化的数据通过增量模式同步到目标表。

增量模式(INC)

全量模式，顾名思议即为对源表增量变化的数据插入目标表，增量模式依赖记录日志功能.

目前增量模式的记录日志功能，是通过oracle的物化视图功能。

创建物化视图。

```
CREATE MATERIALIZED VIEW LOG ON ${tableName} with primary key.
```

· 运行增量模式之前，需要先开启记录日志的功能，即预先创建物化视图. 特别是配合全量模式时，创建物化视图的时间点要早于运行全量之前，这样才可以保证数据能全部同步到目标表

· 增量模式没有完成的概念，它只有追上的概念，具体的停止需有业务进行判断，可以看一下切换流程。

自动模式(ALL)

自动模式，是对全量+增量模式的一种组合，自动化运行，减少操作成本。

自动模式的内部实现步骤：

1. 开启记录日志功能. (创建物化视图)
2. 运行全量同步模式. (全量完成后，自动进入下一步)
3. 运行增量同步模式. (增量模式，没有完成的概念，所以也就不会自动退出，需要业务判断是否可以退出，可以看一下切换流程)

对比模式(CHECK)

对比模式，即为对源库和目标库的数据进行一次全量对比，验证一下迁移结果. 对比模式为一种可选运行，做完全量/增量/自动模式后，可选择性的运行对比模式，来确保本次迁移的正确性.

愚公项目是在阿里去IOE的过程中诞生，其要解决的目标就是**帮助用户完成从Oracle数据迁移到MySQL上.是后续的DTS工具重要组成部分。**经过生产系统的验证，是一种可靠的数据库迁移工具。**关键在于其支持的增量的方式的迁移，能够在数据上云的过程中减少大量的停机时间。**

数据库的迁移不仅是数据迁移，包含的更多技术细节问题。**Bespin Global能够提供大量的数据库迁移的技术支持。**

更多详情
阅读原文

阅读原文