分布式事务解决方案

- 1.引入pom依赖
- 2.分布式事务数据表初始化
    - 2.1 oracle
    - 2.2 postgresql
    - 2.3 mysql
- 3.yaml文件增加配置
- 4.使用样例
- 5.注意事项-AT 模式

> 本文档主要介绍业务中涉及到分布式事务的使用场景，提供了 AT（默认，建议使用，本文主要以 AT为例）、TCC、SAGA 事务模式。

1.引入pom依赖

```
<!--seata start-->
<dependency>
   <groupId>com.acca</groupId>
   <artifactId>araf-distributed-transaction</artifactId>
</dependency>
<!--seata end-->
```

2.分布式事务数据表初始化

根据数据库类型，选择相应的数据库脚本，在各自业务库执行初始化操作，以支持分布式事务

2.1 oracle

```
-- for AT mode you must to init this sql for you business database. the
seata server not need it.
CREATE TABLE undo_log
(
    id            NUMBER(19)    NOT NULL,
    branch_id     NUMBER(19)    NOT NULL,
    xid           VARCHAR2(100) NOT NULL,
    context       VARCHAR2(128) NOT NULL,
    rollback_info BLOB          NOT NULL,
    log_status    NUMBER(10)    NOT NULL,
    log_created   TIMESTAMP(0)  NOT NULL,
    log_modified  TIMESTAMP(0)  NOT NULL,
    PRIMARY KEY (id),
    CONSTRAINT ux_undo_log UNIQUE (xid, branch_id)
);

COMMENT ON TABLE undo_log IS 'AT transaction mode undo table';

-- Generate ID using sequence and trigger
CREATE SEQUENCE UNDO_LOG_SEQ START WITH 1 INCREMENT BY 1;
```

2.2 postgresql

```
-- for AT mode you must to init this sql for you business database. the
seata server not need it.
CREATE TABLE IF NOT EXISTS undo_log
(
    id            SERIAL       NOT NULL,
    branch_id     BIGINT       NOT NULL,
    xid           VARCHAR(100) NOT NULL,
    context       VARCHAR(128) NOT NULL,
    rollback_info BYTEA        NOT NULL,
    log_status    INT          NOT NULL,
    log_created   TIMESTAMP(0) NOT NULL,
    log_modified  TIMESTAMP(0) NOT NULL,
    CONSTRAINT pk_undo_log PRIMARY KEY (id),
    CONSTRAINT ux_undo_log UNIQUE (xid, branch_id)
);

CREATE SEQUENCE IF NOT EXISTS undo_log_id_seq INCREMENT BY 1 MINVALUE 1 ;
```

2.3 mysql

```
-- for AT mode you must to init this sql for you business database. the
seata server not need it.
CREATE TABLE IF NOT EXISTS `undo_log`
(
    `id`            BIGINT(20)   NOT NULL AUTO_INCREMENT COMMENT
'increment id',
    `branch_id`     BIGINT(20)   NOT NULL COMMENT 'branch transaction id',
    `xid`           VARCHAR(100) NOT NULL COMMENT 'global transaction id',
    `context`       VARCHAR(128) NOT NULL COMMENT 'undo_log context,such
as serialization',
    `rollback_info` LONGBLOB     NOT NULL COMMENT 'rollback info',
    `log_status`    INT(11)      NOT NULL COMMENT '0:normal status,1:
defense status',
    `log_created`   DATETIME     NOT NULL COMMENT 'create datetime',
    `log_modified`  DATETIME     NOT NULL COMMENT 'modify datetime',
    PRIMARY KEY (`id`),
    UNIQUE KEY `ux_undo_log` (`xid`, `branch_id`)
) ENGINE = InnoDB
  AUTO_INCREMENT = 1
  DEFAULT CHARSET = utf8 COMMENT ='AT transaction mode undo table';
```

3.yaml文件增加配置

```
# Seata  SeataProperties
seata:
  # Seata  ServiceProperties
  service:
    #
    vgroup-mapping:
      armp-seata-service-group: default # {armp}  spring.application.name,-
seata-service-group
  # Seata  RegistryProperties
  registry:
    type: nacos #
    nacos:
      cluster: default #  Seata
      namespace: ${spring.profiles.active} # Nacos
      serverAddr: 10.1.17.17:8848 # Nacos
```

4.使用样例

```
//@GlobalTransactional


@Override
@GlobalTransactional
public MasCountryVO putTransction(String id, MasCountryVO vo) {
```

5.注意事项-AT 模式

1. 业务表中必须包含单列主键 。
2. 每个业务库中必须包含 undo_log 表，若与分库分表组件联用，分库不分表。
3. 目前AT模式支持的数据库有：MySQL、Oracle、PostgreSQL和 TiDB。
4. 使用注解开启分布式事务时，若默认服务 provider 端加入 consumer 端的事务，provider 可不标注注解。但是，provider 同样需要相应的依赖和配置，仅可省略注解。
5. 使用注解开启分布式事务时，若要求事务回滚，必须将异常抛出到事务的发起方，被事务发起方的 @GlobalTransactional 注解感知到。provide 直接抛出异常 或 定义错误码由 consumer 判断再抛出异常。
6. @Transactional 需要在 @GlobalTransactional 方法里