Application Note
POINT SYSTEM

[목차]

Overview

Application Notes 는 RDLMS(Relational Distributed Ledger Management System)에서 다양한 어플리케이션을 구축하는 방법에 대한 예제를 제공함으로써 개발자들이 쉽게 RDLMS 와 친숙해질 수 있도록 도움을 주는 문서이다.

이 문서는 구체적으로,
발행인이 RDLMS 에서 암호자산 (이하 POINT)을 발행하고, 사용자가 발행된 암호자산을 지불・결제에 이용하는 것을 주 용도로 다음과 같이 3 가지의 어플리케이션을 구현한다.

- 예제 1 에서는 일반적인 블록체인 환경에서와 같이 누구나 무기명으로 참여가 가능한 POINT ANYONE 어플리케이션
- 예제 2 에서는 허가된 자만 사용할 수 있도록 참여가 제한된 POINT MEMBERSHIP 어플리케이션
- 예제 3 은 안전한 거래를 위해 ESCROW (Multisig)이 추가된 POINT ANYONE ESCROW 어플리케이션

이 문서는 RDLMS 가 어떻게 작용하는지에 대한 기본적인 설명을 다루지는 않는다. 유즈케이스 시나리오를 기반으로 전통적인 방식으로 데이터를 디자인하고 여기에 신뢰를 위하여 특정 필드를 추가하거나 테이블의 옵션을 추가하는 방식으로 신뢰할 수 있는 테이블을 어떻게 디자인하고 배포할 것인가에 주력한다. 그리고 예제 1 을 기반으로 어떻게 원장(Table) 을 추가하면서 어플리케이션의 기능을 확장하는지 설명한다.

* 예제 프로그램은 TOSServer, DNOM 과 함께 GIT 을 통해 배포 된다.
GIT ADDRESS : git@github.com:TrustDB/TrustSQLEnv.git

* 예제 프로그램을 포함한 TOSServer, DNOM 의 모든 소스 코드는 다음 주소에서 구할 수 있다.
GIT ADDRESS : git@github.com:TrustDB/TrustSQLEnv.git

* 이 문서에서는 클라이언트, TOSS, DNOM 가 어떻게 통신을 하는지, 그리고 분산원장은 어떻게 배포하는지에 대해서 상세하게 다루지 않는다. 수고스럽더라도 이러한 내용은 소스코드를 참조해야 한다.

1. POINT ANYONE

POINT ANYONE 은 분산원장 기술을 이용하여 POINT 를 발행(MINT)하고, 발행된 POINT 를 이용하여 송금을 수행할 수 있는 기본 어플리케이션이다.

1.1 Usecase Scenario

Actors

- ISSUER
  포인트(가상자산)를 발행하는 주체
- USERs
  ISSUER 가 발생한 포인트를 이용하여 지불/결제를 수행하는 이용자

Usecases

- ***Mint***

Actor	ISSUER
목표	POINT 발행
시작 조건	
이후 조건	발행이 완료되면 발행거래로 기록되어야 함.
정상적인 흐름	ISSUER 가 발행량 결정하여 서명 후 등록
대안 흐름	

- Trasnactions

Actor	USER
목표	자신이 보유한 POINT 를 다른 사람에게 송금
시작 조건	자신이 보유한 POINT 잔액이 0 이상이어야 함.
이후 조건	
정상적인 흐름	USER 가 자신의 잔액 중 특정 양을 다른사람에게 송금
대안 흐름	

1.2 ER(Entity Relationship) Modeling

Entity

- ISSUER :

속성 : None

- USERs :

속성 : None

! Public Transfer Application 에서는 ISSUER, USERs 속성이 없다.

Relations

- 발행(Mint)

속성 : 발행자 번호, 발행량, 발행자 서명

- 송금(Transactions)

속성 : 송금자 주소, 송금액, 수신자 주소, 송금자 서명

Securities

- 모든 레코드는 Actor 에 전자서명을 속성으로 가저야 함.(TRUSTED)
- 모든 레코드는 유일성 보장을 위한 Ordering-Stamp 를 가저야 함.(ORDERED)
- 모든 Table 은 Table Schema 의 무결성을 보장하기 위한 발행자의 서명값을 포함 해야 함.

1.3 Schema Design

1.3.1 MINT Table

- Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN				
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED

TORDER_NUM	BIGINT UNSIGNED	UNIQUE		ORDER
TORDER_NONCE	VARCHAR(16)	UNIQUE	INPUT	
MINTER_TX_ID	VARCHAR(8)	P.K	INPUT	
MINTED_AMOUNT	BIGNT UNSIGNED	N.N	INPUT	
MINTER_SIGN	VARCHAR(160)	N.N	SIGN1)	
TORDER_TIME	DATETIME	N.N		TIME
TORDER_SIGN	VARCHAR(160)	N.N		SIGN2)

SIGN1) INPUTS(TORDER_NONCE,MINTER_TX_ID,MINTED_AMOUNT) VERIFY KEY TABLE_ISSUER

SIGN2) INPUTS(ALL FIELDs exclude TORDER_SIGN) VERIFY KEY TOSA_PUB_KEY

1.3.2 TRANSACTIONS Table

■ Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN	Sign value for Trigger Statement by Issuer.			
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
TORDER_NUM	BIGINT UNSIGNED	P.K		ORDER
TORDER_NONCE	VARCHAR(16)	N.N	INPUT	
MINTER_TX_ID	VARCHAR(8)	F.K, N	INPUT	
SENDER	VARCHAR(66)	N.N	INPUT	
AMOUNT	BIGINT UNSIGNED	N.N	INPUT	
RECEIVER	VARCHAR(66)	N.N	INPUT	
SENDER_SIGN	VARCHAR(160)	N.N	SIGN1)	
TORDER_TIME	DATETIME	N.N		TIME
TORDER_SIGN	VARCHAR(160)			SIGN2)

SIGN1) INPUTS(TORDER_NONCE. MINTER_TX_ID,SENDER_AMOUNT,RECEIVER) VERIFY KEY (SENDER)

SIGN2) INPUTS(ALL FIELDs exclude TORDER_SIGN) VERIFY KEY TOSA_PUB_KEY

* 주소 A 의 Balance 는 다음과 같이 구해진다.

RECEIVE_TOTAL = select sum(AMOUNT) from TRANSACTIONS where RECEIVER='A';

SEND_TOTAL = select sum(AMOUNT) from TRANSACTIONS where SENDER='A'

BALANCE = **RECEIVE_TOTAL** – **SEND_TOTAL**

* TORDER_NONCE

임의로 생성되는 암호화 토큰으로 재생 공격을 방지하기 위한 값. 임의로 생성되는 값 외에 고유한 식별자 (예를 들어 SENDER 의 마지막 트랜잭션의 TORDER_NUM 등)로 사용되는 것이면 모두 가능.

1.3.2.1 Trigger

```
CREATE DEFINER=`www.trustedhotel.com`@`%` TRIGGER
VERIFICATION_TRANSACTIONS_myBit
BEFORE INSERT on TRANSACTIONS_myBit FOR EACH ROW
BEGIN
DECLARE _msg VARCHAR(256);
DECLARE _minted_amount BIGINT DEFAULT 0;
DECLARE _senders_receive_total BIGINT DEFAULT 0;
DECLARE _senders_send_total BIGINT DEFAULT 0;

-- VERIFICATION Rule
-- #1 SENDER, RECEIVER should be different.
IF NEW.SENDER = NEW.RECEIVER THEN
    SET _msg = concat('TRANSACTIONS TABLE VERIFICATION ERROR: SENDER,
RECEIVER SHOULD DIFFERENT!');
signal sqlstate '45000' set message_text = _msg;
END IF;

IF NEW.MINTER_TX_ID IS NOT NULL THEN
    -- #2MINT Transaction
-- NEW.AMOUNT shoud be same MINT_xxxx.MINTED_AMOUNT in SAME KEY(MINTER_TX_ID)
SELECT MINTED_AMOUNT INTO _minted_amount
    FROM MINT_myBit
    WHERE MINTER_TX_ID=NEW.MINTER_TX_ID;
    IF NEW.AMOUNT != _minted_amount THEN
        SET _msg = concat('TRANSACTIONS TABLE VERIFICATION ERROR: In case of
MINT transaction, AMOUNT SHOULD SAME WITH MINT_myBit.MINTED_AMOUNT');
signal sqlstate '45002' set message_text = _msg;
    END IF;
ELSE
```

```
    -- #3 NORMAL Transaction
    -- #AMOUNT SHOUD BE LESS THAN or EQUAL TO SENDER'S BALANCE.

    -- #3-1 SENDER's RECEIVE_TOTAL
    SELECT IFNULL(SUM(AMOUNT),0) INTO _senders_receive_total
    FROM TRANSACTIONS_myBit
    WHERE RECEIVER=NEW.SENDER;

    -- #3-1 SENDER's SEND_TOTAL
    SELECT IFNULL(SUM(AMOUNT),0) INTO _senders_send_total
    FROM TRANSACTIONS_myBit
    WHERE SENDER=NEW.SENDER;

    IF (NEW.AMOUNT > (_senders_receive_total-_senders_send_total)) THEN
        SET _msg = concat('TRANSACTIONS TABLE VERIFICATION ERROR: AMOUNT IS
GREATER THAN BALANCE');
        signal sqlstate '45003' set message_text = _msg;
    END IF;
END IF;
END
```

2. POINT MEMBERSHIP

POINT MEMBERSHIP 은 POINT ANYONE 과 달리 회원(MEMBERSHIP)에 등록된 사용자만 거래에 참여할 수 있다.

2.1 Usecase Scenario

Actors

- ISSUER
  포인트(가상자산)를 발행하는 주체
- Membership
  ISSUER 가 발생한 포인트를 이용하여 지불/결제를 수행하기 위하여 회원가입 된 이용자
- IVA (ID Verificaiton Authority)
  회원가입 된 이용자의 주소인지 검증해 주는 기관

Usecases

- ***Membership***

Actor	IVA (ID Verification Authority)
목표	USER 의 공캐키(주소)를 등록
시작 조건	USER 가 IVA 에 자신을 증명하고 자신의 주소를 등록해 줄것을 요청 한다. (외부 시스템) IVA 는 신원이 확인된 USER 의 주소를 Membership 으로 등록한다.
이후 조건	
정상적인 흐름	IVA 가 USER 의 주소(공개키)에 서명하여 Membership 테이블에 등록
대안 흐름	

- ***Mint***

Actor	ISSUER
목표	POINT 발행
시작 조건	Membership 에 등록된 주소만 거래 가능
이후 조건	발행이 완료되면 발행거래로 기록되어야 함.
정상적인 흐름	ISSUER 가 발행량 결정하여 서명 후 등록
대안 흐름	

- ***Trasnactions***

Actor	USER
목표	자신이 보유한 POINT 를 다른 사람에게 송금
시작 조건	Membership 에 등록된 주소만 거래 가능 자신이 보유한 POINT 잔액이 0 이상이어야 함.
이후 조건	
정상적인 흐름	USER 가 자신의 잔액 중 특정 양을 다른사람에게 송금
대안 흐름	

2.2 ER(Entity Relationship) Modeling

Entity

- ISSUER :

속성 : None

- USERs :

속성 : None

- IVA :

속성 : None

Relations

- 회원등록(Membership)

  속성 : 회원 ID, 주소, 주소에 대한 IVA 의 서명값

- 발행(Mint)

속성 : 발행자 번호, 발행량, 발행자 서명

- 송금(Transactions)

속성 : 송금자 주소, 송금액, 수신자 주소, 송금자 서명

Securities

- 모든 레코드는 Actor 에 전자서명을 속성으로 가저야 함.(TRUSTED)
- 모든 레코드는 유일성 보장을 위한 Ordering-Stamp 를 가저야 함.(ORDERED)
- 모든 Table 은 Table Schema 의 무결성을 보장하기 위한 발행자의 서명값을 포함 해야 함.

2.3 Schema Design

2.3.1 MEMBERSHIP Table

- Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN				
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
TORDER_NUM	BIGINT UNSIGNED	N.N		ORDER
TORDER_NONCE	VARCHAR(16)	N.N	INPUT	
USER_ID	VARCHAR(40)	N.N, UNIQUE	INPUT	
USER_ACCOUNT	VARCHAR(66)	P.K	INPUT	
ISSUER_SIGN	VARCHAR(160)		SIGN1)	
TORDER_TIME	DATETIME			TIME
TORDER_SIGN	VARCHAR(160)			SIGN2)

SIGN1) INPUTS(TORDER_NONCE,USER_ID,USER_ACCOUNT) VERIFY KEY TABLE_ISSUER

SIGN2) INPUTS(ALL FIELDs exclude TORDER_SIGN) VERIFY KEY TOSA_PUB_KEY

2.3.2 MINT Table

■ Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN				
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
TORDER_NUM	BIGINT UNSIGNED	UNIQUE		ORDER
TORDER_NONCE	VARCHAR(16)	UNIQUE	INPUT	
MINTER_TX_ID	VARCHAR(8)	P.K	INPUT	
MINTED_AMOUNT	BIGNT UNSIGNED	N.N	INPUT	

MINTER_SIGN	VARCHAR(160)	N.N	SIGN1)	
TORDER_TIME	DATETIME	N.N		TIME
TORDER_SIGN	VARCHAR(160)	N.N		SIGN2)

SIGN1) INPUTS(TORDER_NONCE,MINTER_TX_ID,MINTED_AMOUNT) VERIFY KEY TABLE_ISSUER
SIGN2) INPUTS(ALL FIELDs exclude TORDER_SIGN) VERIFY KEY TOSA_PUB_KEY

2.3.3 TRANSACTIONS Table

■ Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN	Sign value for Trigger Statement by Issuer.			
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
TORDER_NUM	BIGINT UNSIGNED	P.K		ORDER
TORDER_NONCE	VARCHAR(16)	N.N	INPUT	
MINTER_TX_ID	VARCHAR(8)	F.K1), N	INPUT	
SENDER	VARCHAR(66)	F.K2), N.N	INPUT	
AMOUNT	BIGINT UNSIGNED	N.N	INPUT	
RECEIVER	VARCHAR(66)	F.K3), N.N	INPUT	
SENDER_SIGN	VARCHAR(160)	N.N	SIGN1)	
TORDER_TIME	DATETIME	N.N		TIME
TORDER_SIGN	VARCHAR(160)			SIGN2)

F.K1) CONSTRAINT FOREIGN KEY(MINTER_TX_ID) REFERENCES MINT
F.K2) CONSTRAINT FOREIGN KEY(SENDER) REFERENCES MEMBSERSHIP
F.K3) CONSTRAINT FOREIGN KEY(RECEIVER) REFERENCES MEMBERSHIP
SIGN1) INPUTS(TORDER_NONCE. MINTER_TX_ID,SENDER_AMOUNT,RECEIVER) VERIFY KEY (SENDER)
SIGN2) INPUTS(ALL FIELDs exclude TORDER_SIGN) VERIFY KEY TOSA_PUB_KEY

2.3.4.1 Trigger

- POINT ANYONE 과 동일하다.

3. POINT ANYONE ESCROW

POINT ANYONE ESCROW 는 송금 수행시 ESCROW 를 사용할 수 있는 기능이 추가된 어플리케이션이다.

3.1 Usecase Scenario

Actors

- ISSUER
  포인트(가상자산)를 발행하는 주체
- Membership
  ISSUER 가 발생한 포인트를 이용하여 지불/결제를 수행하기 위하여 회원가입 된 이용자
- ESCROW Agent
  ESCROW 를 위하여 Multisig 거래에 서명하는 기관

Usecases

- ***Mint***

Actor	ISSUER
목표	POINT 발행
시작 조건	Membership 에 등록된 주소만 거래 가능
이후 조건	발행이 완료되면 발행거래로 기록되어야 함.
정상적인 흐름	ISSUER 가 발행량 결정하여 서명 후 등록
대안 흐름	

- ***Trasnactions***

Actor	USER
목표	자신이 보유한 POINT 를 다른 사람에게 송금 ESCROW 를 선택할 수 있음.
시작 조건	Membership 에 등록된 주소만 거래 가능 자신이 보유한 POINT 잔액이 0 이상이어야 함.
이후 조건	
정상적인 흐름	USER 가 자신의 잔액 중 특정 양을 다른사람에게 송금
대안 흐름	

■ ***ESCROW (Multisig)***

Actor	ESCROW Agent
목표	특정 거래조건을 만족한 경우 해당 Transaction 에 다중서명(Multisign)을 생성하여 최종 거래를 확인
시작 조건	Multisig 요청된 거래가 존재 해야 함.
이후 조건	
정상적인 흐름	Multisig 요청된 거래에 한하여 다중서명 생성
대안 흐름	

3.2 ER(Entity Relationship) Modeling

ENTITY

포인트(가상자산)를 생성하는 ISSUER,
생성되는 포인트로 송금을 수행하는 USER(회원)
Multisig 를 수행하는 ESCROW Agent

RELATIONs

포인트 생성 (MINT)
송금 (TRANSACTIONS)
다중서명(MULTISIG)

3.3 Schema Design

3.3.1 MINT Table

■ Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN				
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
MINT_GID	BIGINT UNSIGNED	PRIMARY KEY		ORDER
MINT_NONCE	VARCHAR(16)	UNIQUE	INPUT	

MINT_AMOUNT	BIGNT UNSIGNED	N.N	INPUT	
MINT_SIGN	VARCHAR(160)	N.N	SIGN1)	
TOSA_TIME	DATETIME	N.N		TIME
TOSA_SIGN	VARCHAR(160)	N.N		SIGN2)

SIGN1) INPUTS(MINT_NONCE, MINT_AMOUNT) VERIFY KEY TABLE_ISSUER

SIGN2) INPUTS(ALL FIELDs exclude TOSA_SIGN) VERIFY KEY TOSA_PUB_KEY

3.3.2 TRANSACTIONS Table

■ Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN	Sign value for Trigger Statement by Issuer.			
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
TRANSACTION_GID	BIGINT UNSIGNED	P.K		ORDER
TRANSACTION_NONCE	VARCHAR(16)	N.N	INPUT	
MINT_GID	BIGINT UNSIGNED	F.K, N	INPUT	
SENDER_ACCOUNT	VARCHAR(66)	N.N	INPUT	
AMOUNT	BIGINT UNSIGNED	N.N	INPUT	
RECEIVER_ACCOUNT	VARCHAR(66)	N.N	INPUT	
MULTISIG_YN	ENUM (N,Y)	N.N	INPUT	
MULTISIG_ACCOUNT	VARCHAR(160)	N.N	INPUT	
SENDER_SIGN	VARCHAR(160)		SIGN1)	
TOSA_TIME	DATETIME			TIME
TOSA_SIGN	VARCHAR(160)			SIGN2)

SIGN1) INPUTS(TRANSACTION_NONCE,MINTER_GID, SENDER_AMOUNT,RECEIVER_ACCOUNT,MULTISIG_YN,MULTISIG_ACCOUNT) VERIFY KEY (SENDER_ACCOUNT)

SIGN2) INPUTS(ALL FIELDs exclude TOSA_SIGN) VERIFY KEY TOSA_PUB_KEY

3.3.2.1 Trigger

```
CREATE DEFINER=`www.trustedhotel.com`@`%` TRIGGER VERIFICATION_TRANSACTIONS_myBit
BEFORE INSERT on TRANSACTIONS_myBit FOR EACH ROW
BEGIN
DECLARE _msg VARCHAR(256);
DECLARE _minted_amount BIGINT DEFAULT 0;
DECLARE _senders_receive_total BIGINT DEFAULT 0;
DECLARE _senders_send_total BIGINT DEFAULT 0;
DECLARE _multis_senders_receive_total BIGINT DEFAULT 0;
DECLARE _multis_senders_send_total BIGINT DEFAULT 0;
DECLARE _balance BIGINT DEFAULT 0;

-- VERIFICATION Rule
-- #1 SENDER, RECEIVER should be different.
IF NEW.SENDER_ACCOUNT = NEW.RECEIVER_ACCOUNT THEN
SET _msg = concat('TRANSACTIONS TABLE VERIFICATION ERROR: SENDER, RECEIVER SHOULD
DIFFERENT!');
signal sqlstate '45000' set message_text = _msg;
END IF;

IF NEW.MINT_GID IS NOT NULL THEN
-- #2MINT Transaction
-- NEW.AMOUNT shoud be same MINT_xxxx.MINTED_AMOUNT in SAME KEY(MINTER_TX_ID)
SELECT MINT_AMOUNT INTO _minted_amount FROM MINT_myBit WHERE MINT_GID=NEW.MINT_GID;

IF NEW.AMOUNT != _minted_amount THEN
SET _msg = concat('TRANSACTIONS TABLE VERIFICATION ERROR: In case of MINT transaction,
AMOUNT SHOULD SAME WITH MINT_myBit.MINT_AMOUNT');
signal sqlstate '45002' set message_text = _msg;
END IF;
ELSE
-- #3 NORMAL Transaction
-- #AMOUNT SHOUD BE LESS THAN or EQUAL TO SENDER'S BALANCE. p2p

-- #3-1 SENDER's RECEIVE_TOTAL
SELECT IFNULL(SUM(AMOUNT),0) INTO _senders_receive_total
FROM TRANSACTIONS_myBit
WHERE RECEIVER_ACCOUNT=NEW.SENDER_ACCOUNT AND MULTISIG_YN='N';

-- #3-2 SENDER's SEND_TOTAL
SELECT IFNULL(SUM(AMOUNT),0) INTO _senders_send_total
FROM TRANSACTIONS_myBit
WHERE SENDER_ACCOUNT=NEW.SENDER_ACCOUNT AND MULTISIG_YN='N';

-- #4-1 MULTISIG SENDER's RECEIVE_TOTAL
SELECT IFNULL(SUM(a.AMOUNT),0) INTO _multis_senders_receive_total
FROM TRANSACTIONS_myBit AS a, MULTISIG_myBit AS b
WHERE a.RECEIVER_ACCOUNT=NEW.SENDER_ACCOUNT AND a.MULTISIG_YN='Y' AND
b.TRANSACTION_GID=NEW.TRANSACTION_GID;

-- #4-2 MULTISIG SENDER's SEND_TOTAL
SELECT IFNULL(SUM(a.AMOUNT),0) INTO _multis_senders_send_total
FROM TRANSACTIONS_myBit AS a, MULTISIG_myBit AS b
WHERE a.SENDER_ACCOUNT=NEW.SENDER_ACCOUNT AND a.MULTISIG_YN='Y';

SET _balance = (_senders_receive_total+_multis_senders_receive_total) -
(_senders_send_total+_multis_senders_send_total);

IF (NEW.AMOUNT > _balance) THEN
SET _msg = concat('TRANSACTIONS TABLE VERIFICATION ERROR: AMOUNT IS GREATER THAN
BALANCE');
signal sqlstate '45003' set message_text = _msg;
END IF;
END IF;
END
```

3.3.3 MULTISIG Table

■ Table Schema

Table Options				
TRUSTED_TYPE	TRUSTED, ORDERED			
DSA_SCHEME	SECP256K1			
TABLE_ISSUER_PUB_KEY	Issuer's public key (compressed)			
TOSA_PUB_KEY	TOSA's public key			
TRIGGER_BEFORE_INSERT_SIGN				
Fields				
Name	Data Type	Const.	TRUSTED	ORDERED
MULTISIG_GID	BIGINT UNSIGNED	UNIQUE		ORDER
MULTISIG_NONCE	VARCHAR(16)	UNIQUE	INPUT	
TRANSACTION_GID	VARCHAR(8)	P.K	INPUT	
MULTISIG_ACCOUNT		N.N	INPUT	
MULTISIG_SIGN	VARCHAR(66)	N.N	SIGN1)	
TOSA_TIME	DATETIME	N.N		TIME
TOSA_SIGN	VARCHAR(160)	N.N		SIGN2)

SIGN1) INPUTS(MULTISIG_NONCE,TRANSACTION_GID,MULTISIG_ACCOUNT) VERIFY KEY (MULTISIG_ACCOUNT)

SIGN2) INPUTS(ALL FIELDs exclude TORDER_SIGN) VERIFY KEY TOSA_PUB_KEY

! MULTISIG 계정을 Table Owner 의 것으로 고정할 수 도 있고, 트랜잭션별로 다르게 설정할 수 도 있다. 여기서는 트랜잭션별로 다르게 설정하도록 한다.

4. Global Ledger ID

각각의 어플리케이션은 DLN (Distributed Ledger Network)상에서 배포되기 위하여 유일한 식별체제가 필요하다. 다음과 같은 형태로 구성한다.

GSID(Global Service ID)

[Issuer ID]/[Service ID]

GLID(Global Ledger ID)

[GSID]/[Ledger ID]

발행자이름을 www.trusted.com, 서비스이름을 application 이름으로 한다면 다음과 같은 GLID 를 생성할 수 있다.

ISSUER	Service	Table	GLID
www.trustedhotel.com	point_anyone	mint	www.trustedhotel.com/point_anyone/mint
		transactions	www.trustedhotel.com/point_anyone/transactions
	point_membership	membership	www.trustedhotel.com/point_membership/membership
		mint	www.trustedhotel.com/point_membership/mint
		transactions	www.trustedhotel.com/point_membership/transactions
	point_escrow	mint	www.trustedhotel.com/point_escrow/mint
		transactions	www.trustedhotel.com/point_escrow/transactions'
		multisig	www.trustedhotel.com/point_escrow/multisig

* Issuer ID 는 database 의 account, Service ID 는 database 의 DB, Ledger ID 는 database 의 Table ID 로 매칭 된다.

5. Application Deployment

5.1 DLN(Distributed Ledger Network)

DLN 에는 2 가지 타입의 노드가 존재한다.

TOSS (Trusted Ordering Stamping Server)

거래의 단일 수집 창구로, 전체 순번을 관리하고 순번을 보증하고 거래를 분산노드에 전파하는 서비스를 수행하는 서버

DNODE (Distributed NODE) (DNOM)

분산원장을 저장하고 서비스하는 다수의 서버

5 장에서는 하나의 TOSS 에 4 개의 DNODE 를 구성하여 서비스를 운영하는 것을 목표로 어떻게 네트워크를 구성하고 배포해야 하는지를 설명한다.

5.1.1 Configuration for Network

각 노드들은 TCP/IP 네트웍 통신을 수행하므로 IP Address 와 Port 의 정보가 필요하다.
본 예제에서는 DLN 을 최대한 간략하게 구축하고자 하므로 모두 동일한 서버에서 동작하도록 설정 하였다. (실제 물리적 구성 시 각 노드들은 안전을 위하여 최대한 분리된 네트워크에 존재하는 것이 좋다)

TYPE	IP	Port	Description
TOSS	localhost	20001	Publish Service 를 위한 Port
	localhost	20002	Subscribe Service 를 위한 Port
DNOM_1	localhost	10001	DNOM Service 를 위한 Port
	localhost	3306	DB Port
DNOM_2	localhost	10002	DNOM Service 를 위한 Port
		3316	DB Port
DNOM_3	localhost	10003	DNOM Service 를 위한 Port
		3326	DB Port
DNOM_4	localhost	10004	DNOM Service 를 위한 Port
		3336	DB Port

5.1.2 configuration for KEY

TOSS 에는 트랜잭션에 Orderig-Stamp 를 생성하기 위한 서명 작업을 수행하므로 TOSA's Key 한쌍 (Public Key, Private)이 모두 필요하다.
DNOM 은 TOSS 가 생성한 Ordering-Stamp 를 검증해야 하므로 TOSA's Public Key 가 필요하다. 따라서 각각 필요한 키를 각 서버에 배치시켜야 한다. TrustSQLEnv 배포버전은 키 생성 툴(wallet)을 포함하고 있다. 이 툴을 이용해야 한다.

5.1.3 configuration for TOSS

다음은 TOSS 구성 파일의 예이다. 각 값의 의미는 사용자 매뉴얼을 참조하라.

```
{
	"TOSSParameters":
	{
		"publish_service_port":"20001",
		"subscribe_service_port":"20002",
		"dataFolderPath":"./raw_transactions",
		"dataFileSizeMB":"256",
		"transactionDownloadLimit":"5000"
	},
	"ServiceConfig":
	[
		{
			"serviceName":"www.trustedhotel.com/point_anyone",
			"keyFileNamePath":"key_toss_00001",	<- TOSA's Key for Ordering Stamp for Service
			"whitelist_node":
			[
				{"addr":"127.0.0.1"},
			]
		},
		{
			"serviceName":"www.trustedhotel.com/point_anyone_escrow",
			"keyFileNamePath":"key_toss_00001",
			"whitelist_node":
			[
				{"addr":"127.0.0.1"},
			]
		},
		{
			"serviceName":"www.trustedhotel.com/point_membership",
			"keyFileNamePath":"key_toss_00001",
			"whitelist_node":
			[
				{"addr":"127.0.0.1"},
			]
		},
		{
			"serviceName":"www.trustedhotel.com/membership",
			"keyFileNamePath":"key_toss_00001",
			"whitelist_node":
			[
				{"addr":"127.0.0.1"},
			]
		},
		{
			"serviceName":"www.trustedhotel.com/nft_voucher",
			"keyFileNamePath":"key_toss_00001",
			"whitelist_node":
			[
				{"addr":"127.0.0.1"},
			]
		},

	]
}
```

5.1.4 configuration for DNOM

다음은 dnom 구성 파일의 예이다. 각 값의 의미는 사용자 매뉴얼을 참조하라.

```
{
	"PublicDNOMParameters":
	{
		"dataFolderPath":"./raw_transactions",
		"dataFileSizeMB":"256",
		"transactionDownloadLimit":"50000",
		"dnomServicePort":"10001",
		"publisherSleepTime":"100",
		"subscriberSleepTime":"10"
	},
	"ServiceConfig":
	[
		{
			"serviceName":"www.trustedhotel.com/point_anyone",
			"tossAddrs":"localhost:20001",
			"rdlmsAddrs":"jdbc:mariadb://127.0.0.1:3306"
		},
		{
			"serviceName":"www.trustedhotel.com/point_anyone_escrow",
			"tossAddrs":"localhost:20001",
			"rdlmsAddrs":"jdbc:mariadb://127.0.0.1:3306"
		},
		{
			"serviceName":"www.trustedhotel.com/point_membership",
			"tossAddrs":"localhost:20001",
			"rdlmsAddrs":"jdbc:mariadb://127.0.0.1:3306"
		},
		{
			"serviceName":"www.trustedhotel.com/membership",
			"tossAddrs":"localhost:20001",
			"rdlmsAddrs":"jdbc:mariadb://127.0.0.1:3306"
		},
		{
			"serviceName":"www.trustedhotel.com/nft_voucher",
			"tossAddrs":"localhost:20001",
			"rdlmsAddrs":"jdbc:mariadb://127.0.0.1:3306"
		}

	]
}
```

! 붉은색 값은 DNOM 서버에 따라 바뀌는 것들이다. 5.1.1 을 참조하여 적절한 값을 기입해야 한다.

6. Client Application

예제에서는 Client 에서 배포된 분산원장을 이용하여 트랜잭션을 조회 및 생성하는 어플리케이션을 제공한다. 다음과 같은 기능을 수행한다. 이 예제는 rdlms 배포 패키지에 포함되어 있다.

```
$ ./demo
#################################################################
#    Copyright © 2021.11 TrustDB ( www.trust-db.com )           #
#      All rights reserved                                      #
#                                                               #
#    DEMO Launcher                                              #
#                                                               #
#    Create Date : 2021/11/14                                   #
#################################################################

================================================================
Demo Application List
================================================================
[NFT VOUCHER]
1 Mebership Management
2 Deploying NFT Voucher & Minting
3 Deploying NFT Voucher Trsnactions & Making Transactions

[POINT ANYONE]
4 Deploying POINT & Minting
5 Making Transactions

[POINT MEMBERSHIP]
6 Deploying POINT & MEMBERSHIP, & Minting
7 Making Transactions

[POINT ANYONE ESCROW]
8 Deploying POINT, ESCROW & MEMBERSHIP, & Minting
9 Making Transations
----------------------------------------------------------------
[USAGE]./demo no
```

! 이 문서는 demo Application 중 [NFT Voucher] 에 대한 설명은 다루지 않는다. NFT Voucnher 는 예제 1, 2, 3 을 활용하여 구현가능하다.