Target of the project

revision 1

4 August 2019

本資料について

- BBc-1トランザクションのデータモデルをグラフデータベースで扱えるようにするための要件を整理する

- 作成日：2019/8/4
- 作成者：takeshi@quvox.net (t-kubo@zettant.com)

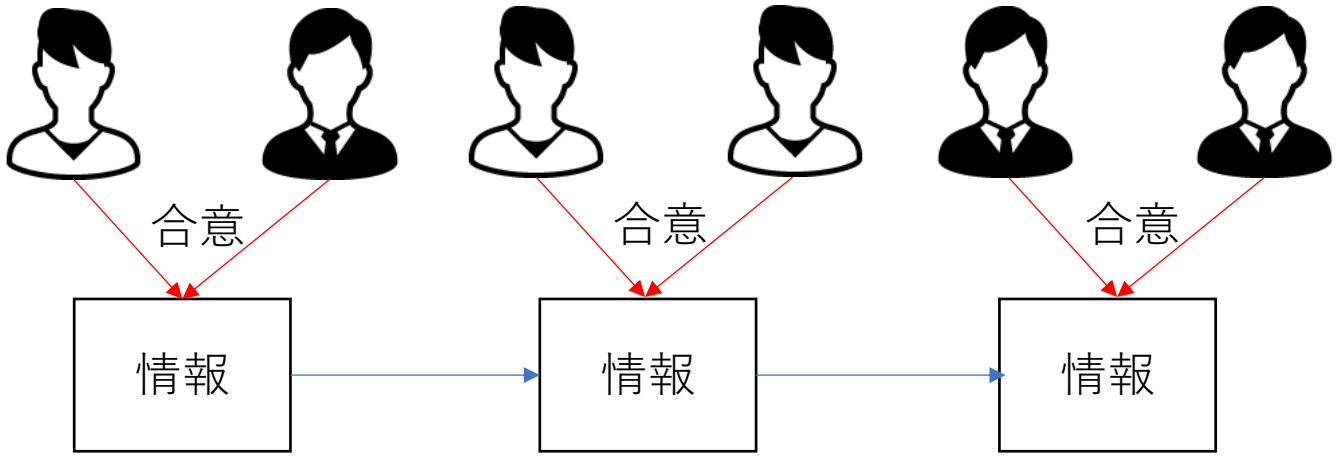
目次

タイトル	ページ
BBc-1の対象と課題	4
トランザクションの構造とグラフ構造	12
システム構成	17
Neo4jへの適用	22

BBc-1の対象と課題

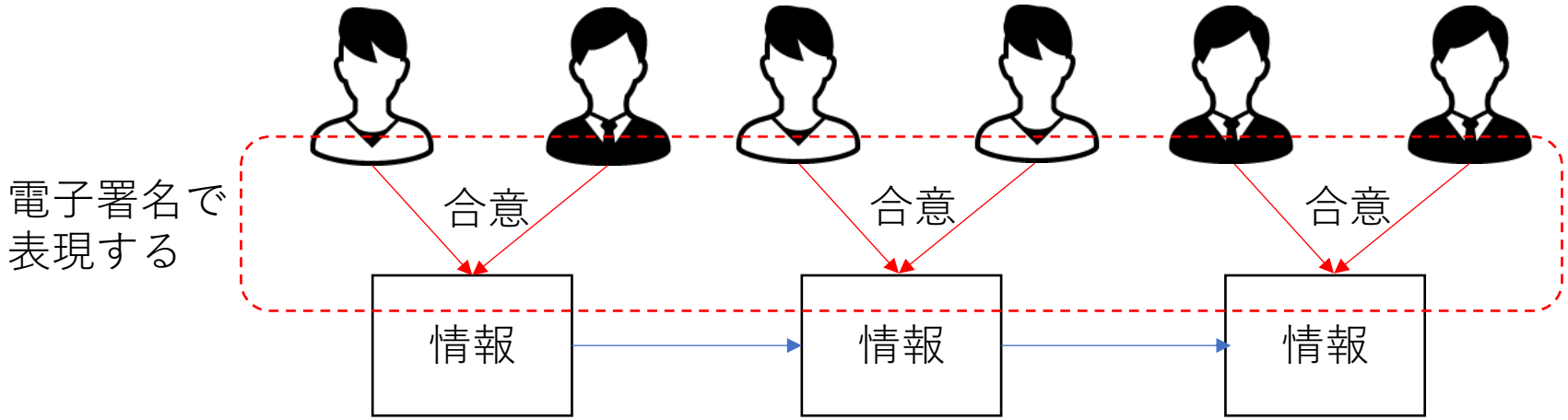
BBc-1を利用する動機

- なんらかの「情報」に対して、ビジネス上の合意を表現し、その「合意したという事実」を覆せないようにする
- さらに、そのような情報同士の関係性についても記録し、その関係性自体への改ざんも防ぎたい



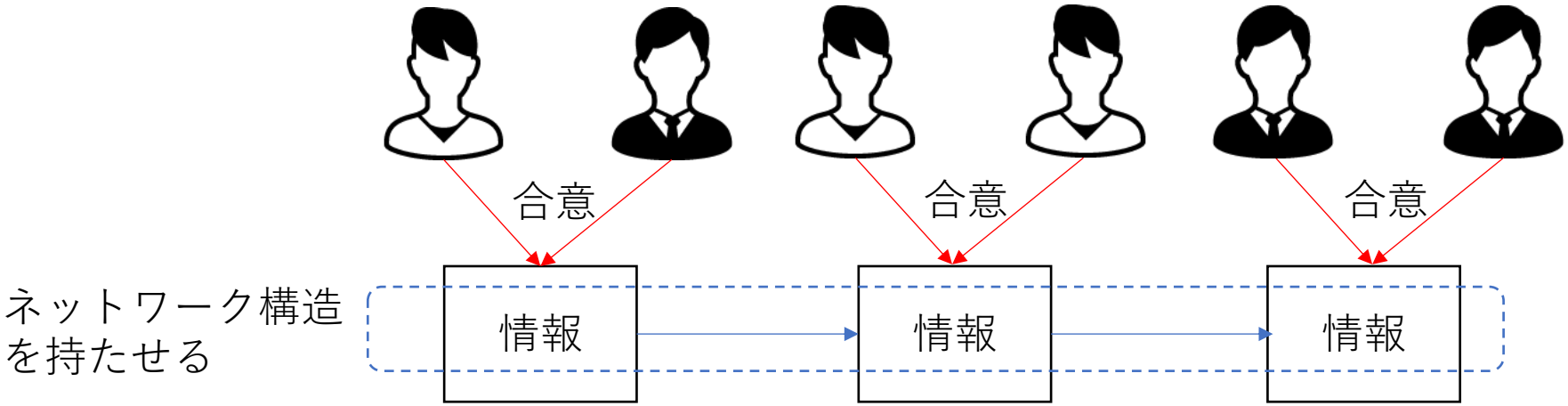
BBc-1を利用する動機

- なんらかの「情報」に対して、ビジネス上の合意を表現し、その「合意したという事実」を覆せないようにする
- さらに、そのような情報同士の関係性についても記録し、その関係性自体への改ざんも防ぎたい



BBc-1を利用する動機

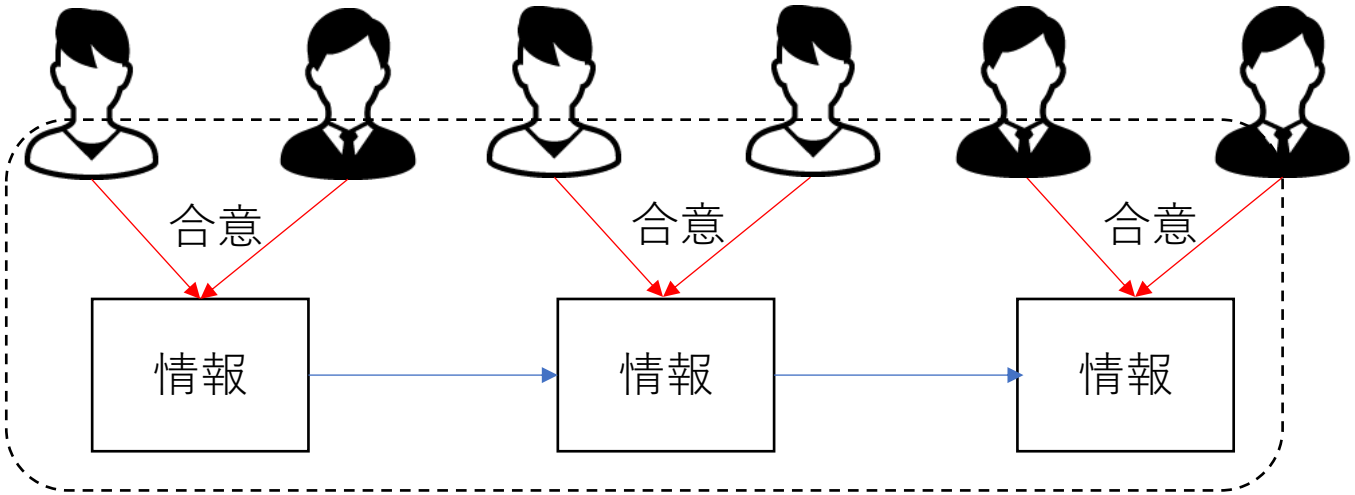
- なんらかの「情報」に対して、ビジネス上の合意を表現し、その「合意したという事実」を覆せないようにする
- さらに、そのような情報同士の関係性についても記録し、その関係性自体への改ざんも防ぎたい



BBc-1を利用する動機

- なんらかの「情報」に対して、ビジネス上の合意を表現し、その「合意したという事実」を覆せないようにする
- さらに、そのような情報同士の関係性についても記録し、その関係性自体への改ざんも防ぎたい

電子署名によって
改ざんから守る



BBc-1 (bbc_core)の課題

- トランザクション情報の検索機能が貧弱
  - 比較的単純な検索機能しか無い（それで十分なのかもしれないが）
  - DBのパフォーマンスチューニングを全く行っていない
    - ノード間でのデータ複製も自前の処理で行っている
    - 署名検証なども行っているので、単純なDBのレプリケーションとは異なる
  - BBc-1独自の考え方で検索する必要がある

メジャーな仕組みを利用できるようにした方が
敷居が下がり、裾野も広がるのでは？

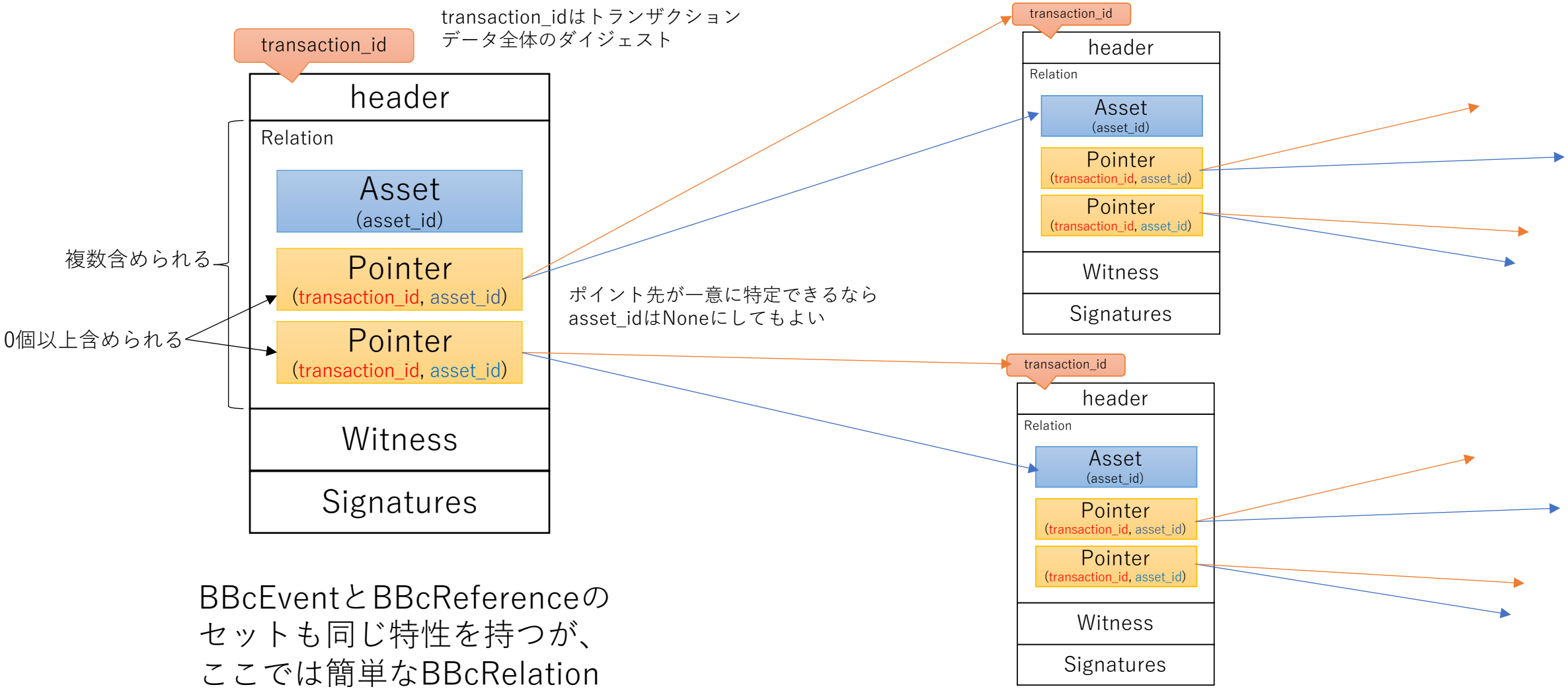
グラフデータベースに適用したい

- トランザクション群はネットワーク構造を持つ
  - トランザクション情報をノード
  - トランザクションから別のトランザクションへのポインタ（参照関係）をエッジ

- トランザクションをそのままグラフDBに投入し、検索できるようにすれば、はやりの技術に乗ることができる
  - とりあえず有名所でneo4jを選択

トランザクションの構造とグラフ構造

トランザクションの構造



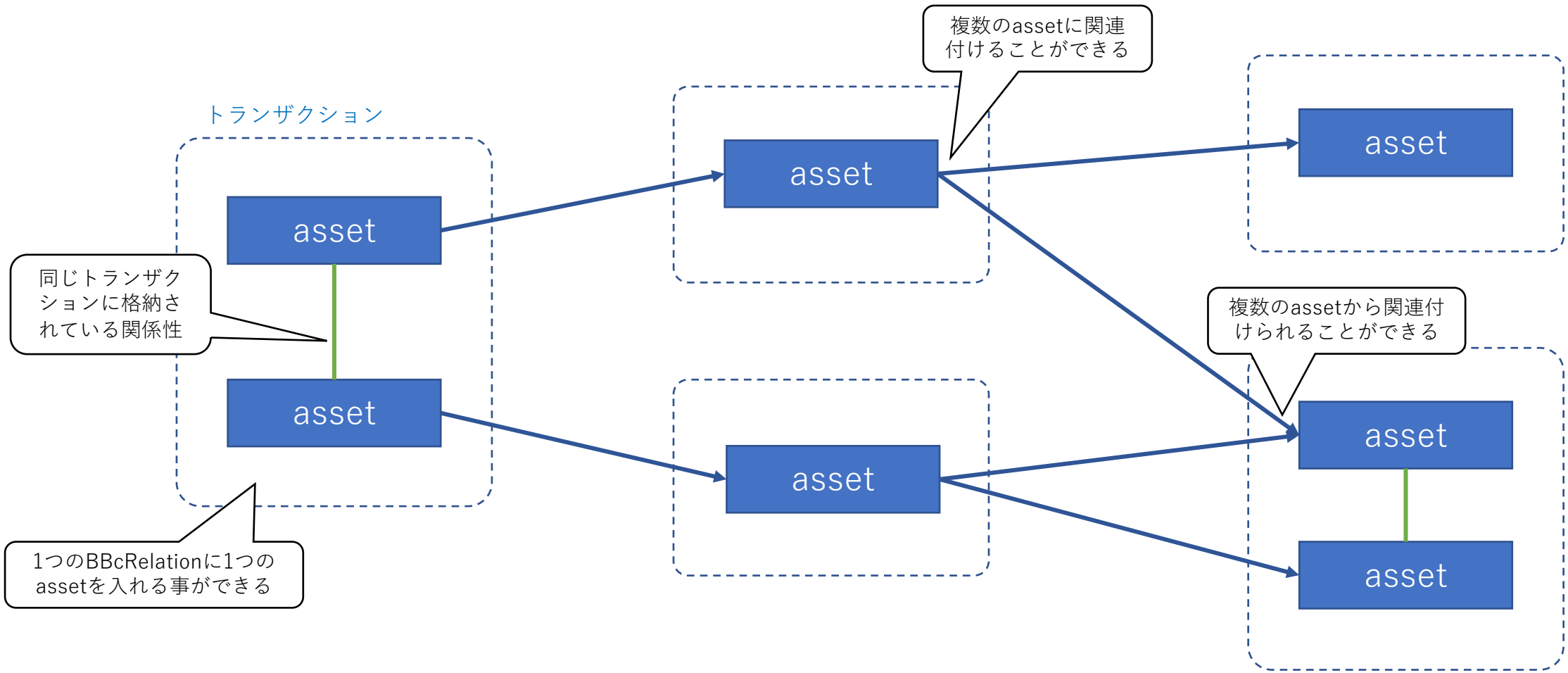
BBcEventとBBcReferenceの
セットも同じ特性を持つが、
ここでは簡単なBBcRelation
を取り上げて検討する

考え方

- 基本は、BBc1のトランザクションは「アセット同士の関係性を表現している」と考える

- アセット同士の関係性の種類
  - 同じトランザクションに含まれている、という関係性
  - ポインタで指定されているという関係性

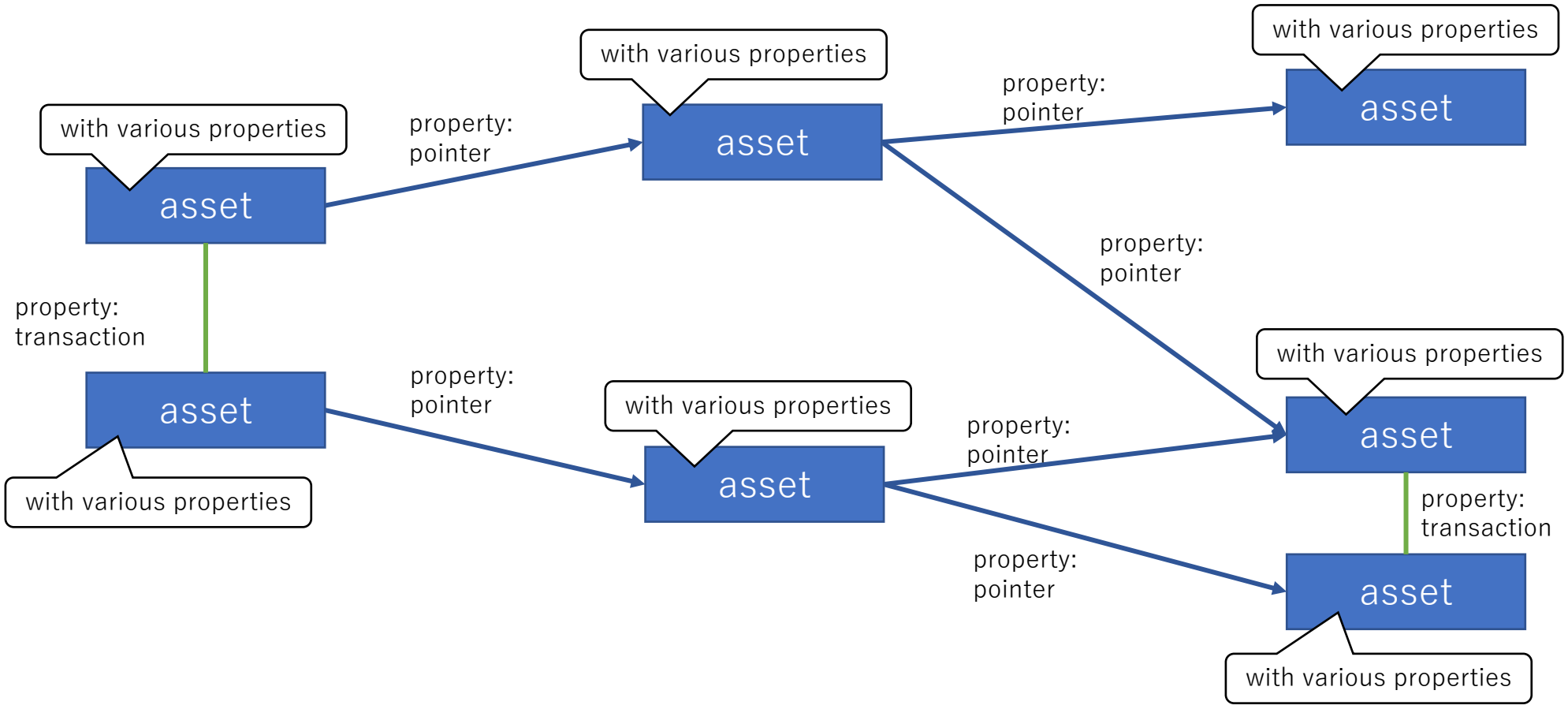
抽象化



グラフ構造への適用

- ノード
  - アセット
    - ノード属性としてアセット情報をもつ
- エッジ
  - ポインタで指定されている
  - または同じトランザクションに含まれている
    - エッジ属性としてポインタまたは同一トランザクションの種別を持つ

グラフ構造



システム構成

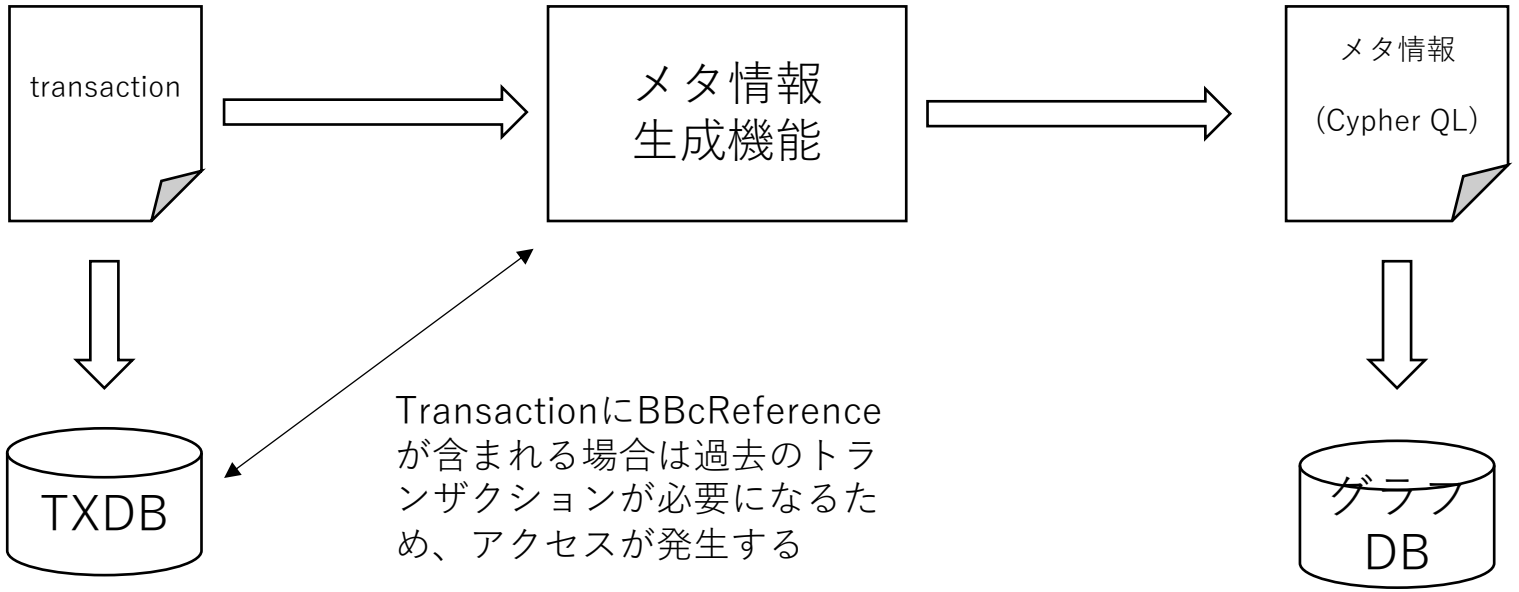
データベースの構成

- グラフDBはトランザクション本体は管理せず、メタ情報のみを管理する
  - 大きなバイナリデータとグラフDBの相性が悪い
    - アンチパターンになっている
    - https://neo4j.com/blog/dark-side-neo4j-worst-practices/
  - トランザクションが改ざんされていないかどうかはグラフDBでは判断できないので、トランザクション本体を検証するしかない
  - メタ情報はトランザクション本体からいつでも（誰でも）作成・再生できる
    - 作成するときに署名検証を行うこと
  - メタ情報にどこまでの情報を含めるかは実装次第
    - 方針は必要

- トランザクション本体はDB（TXDB）に格納する
  - transaction_idとtransaction_dataの紐付けだけでいいので、性能重視のDB実装を利用すればいい
  - TXDBだけはしっかり保全しなければならない

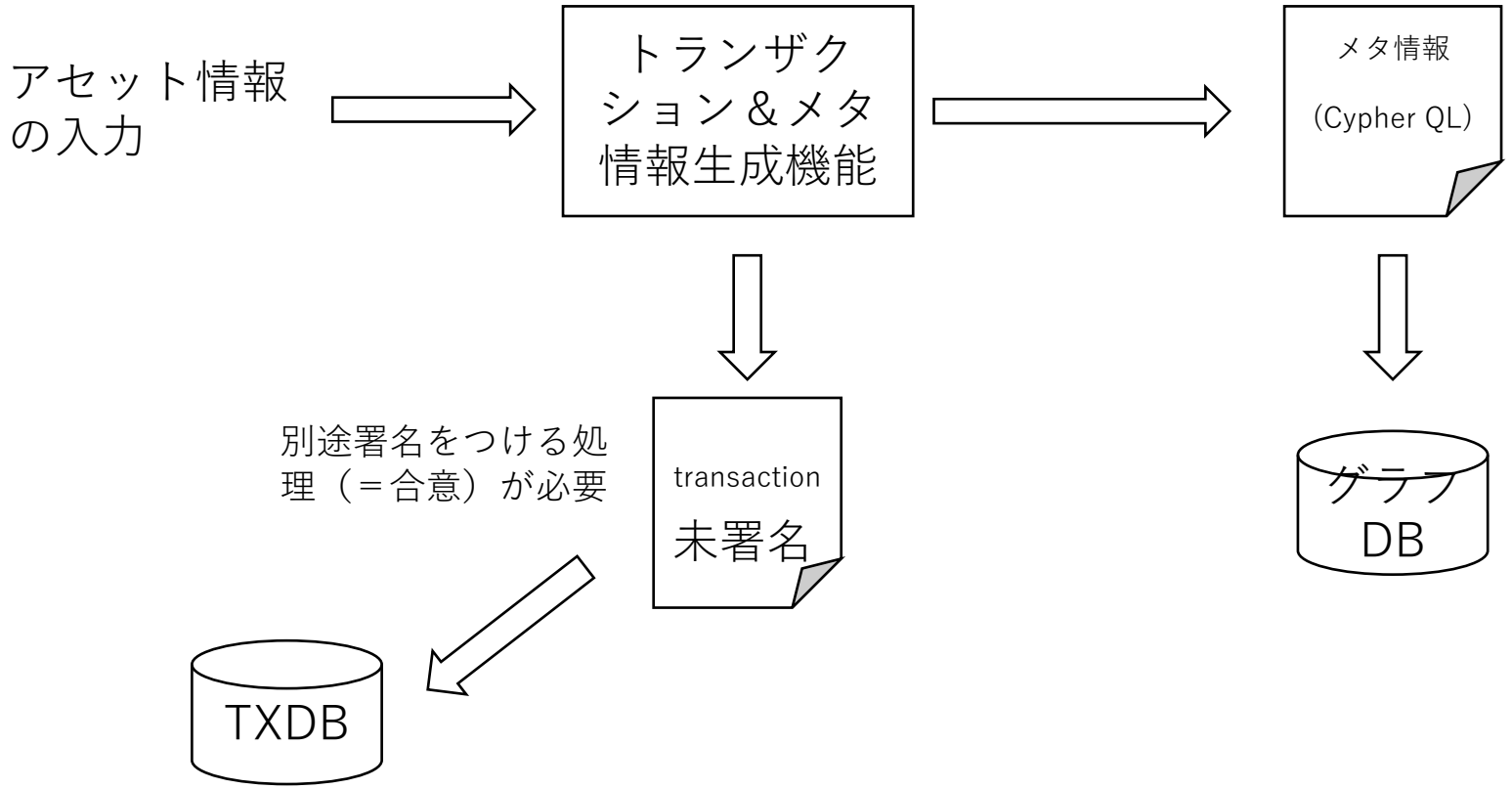
メタ情報生成の実現パターン１

すでに出来上がっている
トランザクション



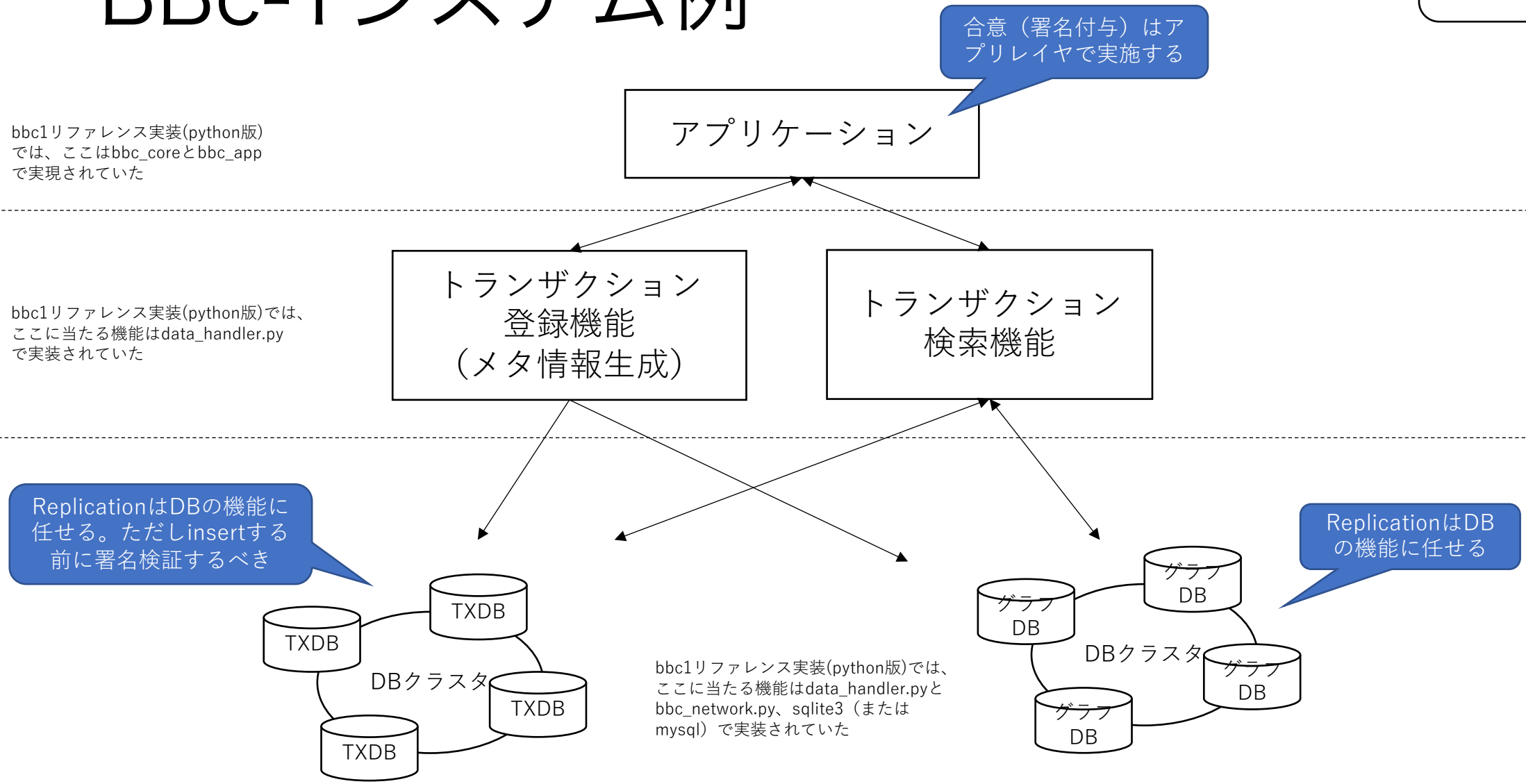
TXDBからトランザクションを読み込んで
メタ情報を生成（再生）することも可能

メタ情報生成の実現パターン２



BBc-1システム例

これらすべての機能については、どこにホストするかは自由に選べばよい



Neo4jへの適用

Neo4j

- オープンソースでは最も人気があるグラフデータベース
  - https://neo4j.com

- 本体はJava実装だが、Boltプロトコルという仕様があり、それを他言語で実装したBoltドライバが公開されている
  - .NET/Java/JavaScript/Python/Goなどが公式にサポートされている

- Cypherクエリ言語（CypherQL）で検索できる

- ノードやエッジにプロパティを設定できる
  - すべてJSON形式

Neo4jに対応するために

- BBcAssetのasset_bodyには、JSON文字列を記述するものとする
  - そのままそれをグラフのノードのプロパティとすることで、asset_bodyの内容でも検索できるようになる

- Neo4jでは、transaction_idなど各種IDはバイナリではなく16進文字列化してプロパティとして登録する
  - Pros: 可視化しやすくなる。実装しやすくなる
  - Cons: データサイズが2倍になる

以上