

BBc-1 system design guide

revision 1

20 April 2019

本資料について

- BBc-1を用いたシステムの設計指針についてまとめる
 - ここで示した指針は完成したものではないため、この資料がさらなる議論のきっかけになることを期待する
- 作成日：2019/4/20
- 作成者：takeshi@quvox.net (t-kubo@zettant.com)

目次

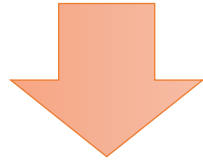
タイトル	ページ
ブロックチェーンそもそも論	4
要件の検討	17
BBc-1によるシステム化	24

ブロックチェーンそもそも論

ビットコインの仕組みは？

- 特定の参加者に全面的な信頼を置きたくない
- でも2重支払いなどの不正を防止できるようにしたい

Ethereumは支払いだけでなくプログラム（スマートコントラクト）実行結果の不正を防止したい



だから、分散システムを仮定せざるを得ない

だから、ビザンチン障害を仮定せざるを得ない

だから、全参加者で確認で移転前後の総量を確認せざるを得ない

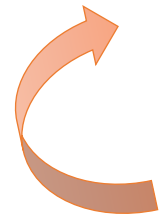
要件の整理が必要

- 不特定多数が参加し誰も信頼しないけど、価値のやり取りをしたい
→ パブリックブロックチェーンを使えばいい
- 特定の参加者が参加し、フラットな信頼関係で情報を管理したい
→ プライベート/コンソーシアム ブロックチェーンを使えばいい

要件の整理が必要

- 不特定多数が参加し誰も信頼しないけど、価値のやり取りをしたい
→ パブリックブロックチェーンを使えばいい

- 特定の参加者が参加し、フラットな信頼関係で情報を管理したい
→ プライベート/コンソーシアム ブロックチェーンを使えばいい



BBc-1は基本的にはプライベート/コンソーシアム型のプラットフォームなので、後者をターゲットとする

そもそも論

- ブロックチェーンを何のために使うのか
 - 情報が「改ざんされていないこと」を確認できるようにする
 - 情報が「存在していたこと」を否定できないようにする
 - 上記2つを誰でも実施できるようにする
- これを実現するのに、必ずしも分散システムである必要はない

実現方法

- ブロックチェーンを何のために使うのか

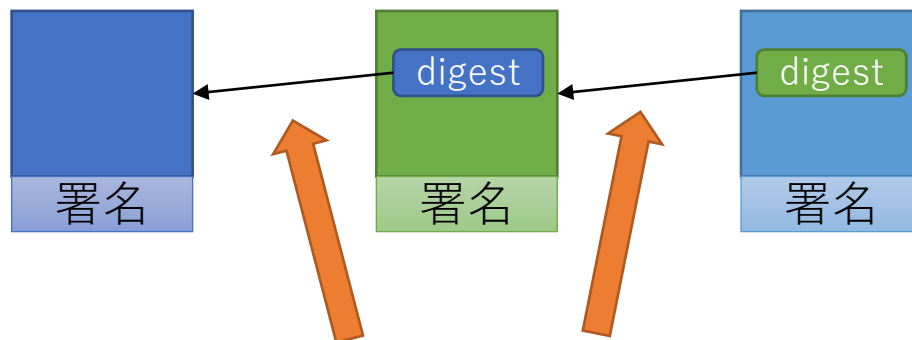
- 情報が「改ざんされていないこと」を確認できるようにする
- 情報が「存在していたこと」を否定できないようにする
- 上記2つを誰でも実施できるようにする

署名がついていれば
改ざんは検知できる



実現方法

- ブロックチェーンを何のために使うのか
 - 情報が「改ざんされていないこと」を確認できるようにする
 - 情報が「存在していたこと」を否定できないようにする
 - 上記2つを誰でも実施できるようにする

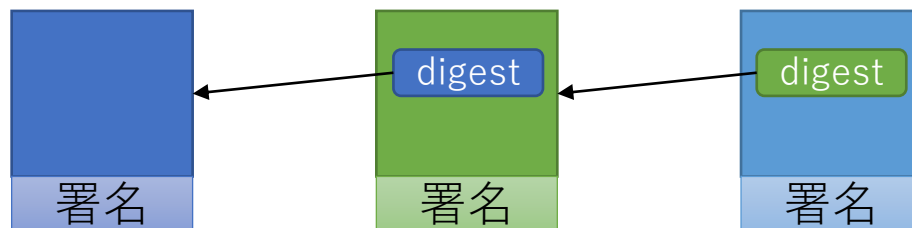


過去の情報のダイジェストを含めた、いわゆる「DAG構造」を構成すれば、途中の情報が削除されるとそれを検知できる

実現方法

- ブロックチェーンを何のために使うのか

- 情報が「改ざんされていないこと」を確認できるようにする
- 情報が「存在していたこと」を否定できないようにする
- 上記2つを誰でも実施できるようにする

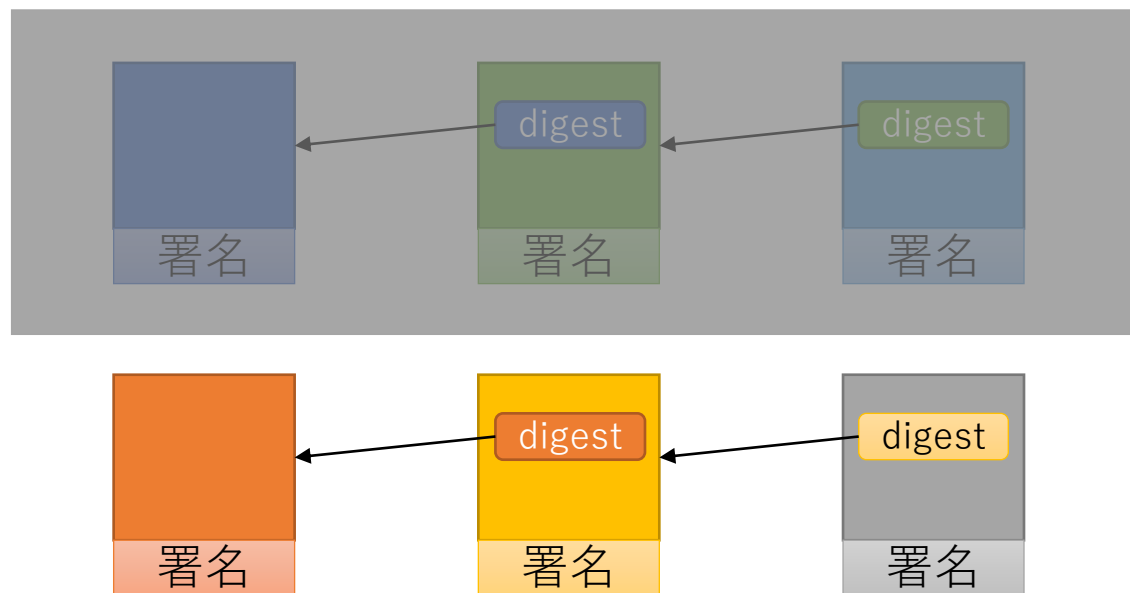


ただし、これだけでは防げない事がある！

実現方法

- ブロックチェーンを何のために使うのか

- 情報が「改ざんされていないこと」を確認できるようにする
- 情報が「存在していたこと」を否定できないようにする
- 上記2つを誰でも実施できるようにする

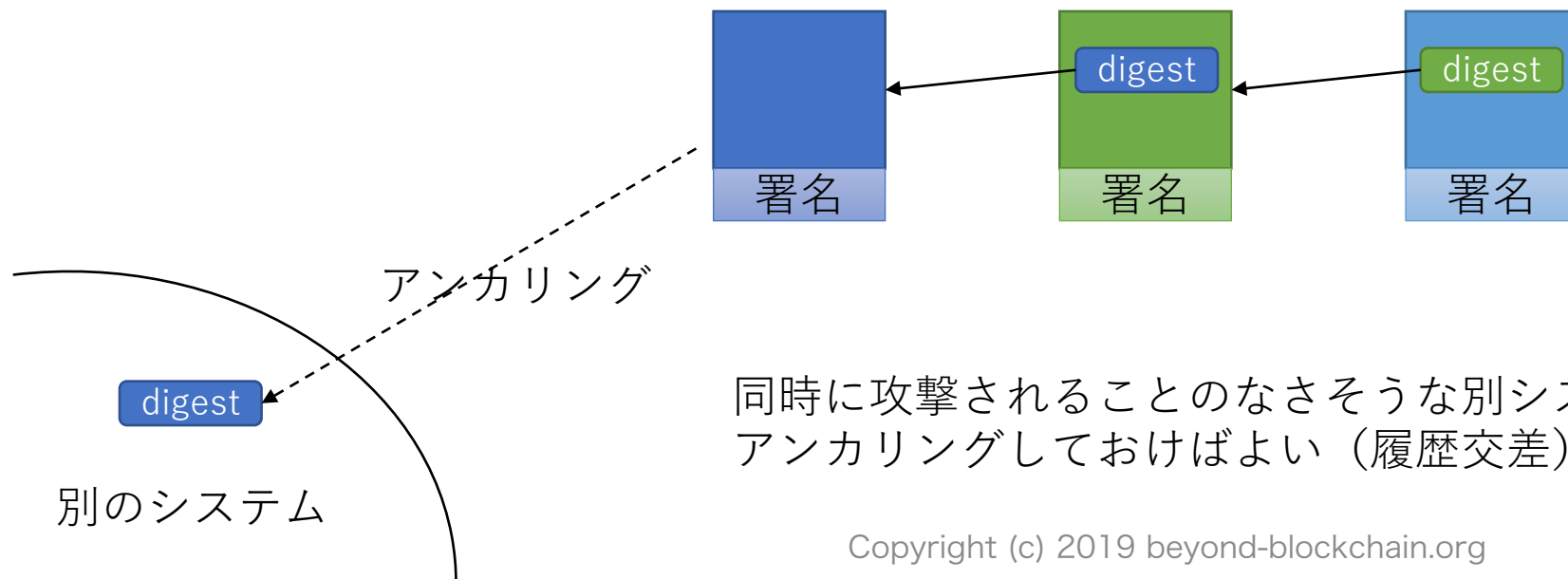


全体をまるごと全部
捏造されたら??

実現方法

- ブロックチェーンを何のために使うのか

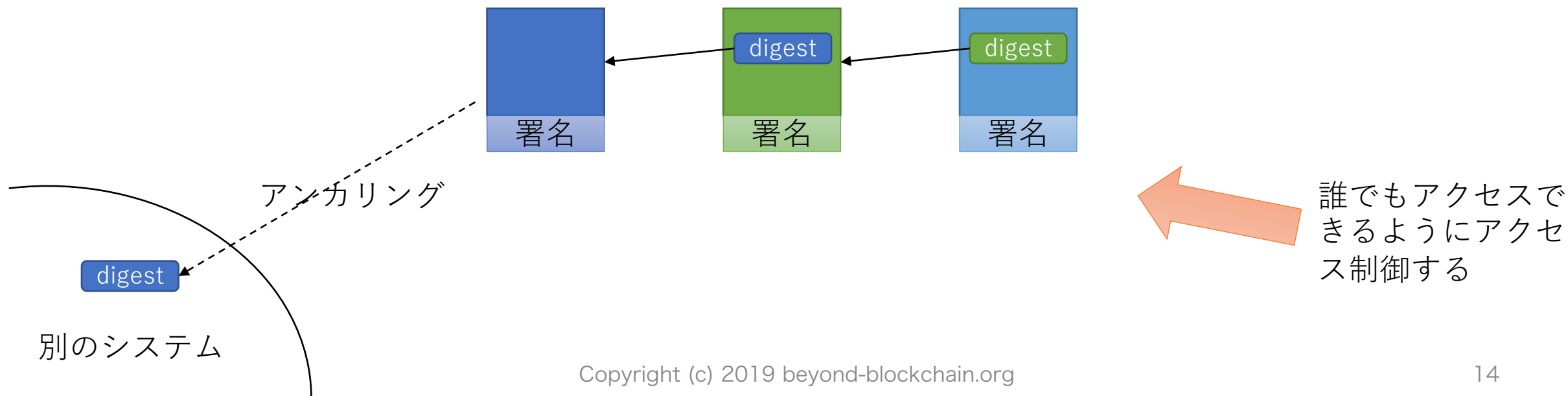
- 情報が「改ざんされていないこと」を確認できるようにする
- 情報が「存在していたこと」を否定できないようにする
- 上記2つを誰でも実施できるようにする



同時に攻撃されることのなさそうな別システムに一部のダイジェストをアンカリングしておけばよい（履歴交差）

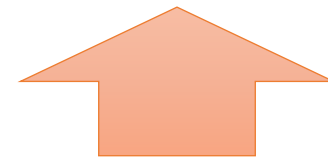
実現方法

- ブロックチェーンを何のために使うのか
 - 情報が「改ざんされていないこと」を確認できるようにする
 - 情報が「存在していたこと」を否定できないようにする
 - 上記2つを誰でも実施できるようにする



そもそも論（もう一度）

- ブロックチェーンを何のために使うのか
 - 情報が「改ざんされていないこと」を確認できるようにする
 - 情報が「存在していたこと」を否定できないようにする
 - 上記2つを誰でも実施できるようにする
- これを実現するのに、必ずしも分散システムである必要はない

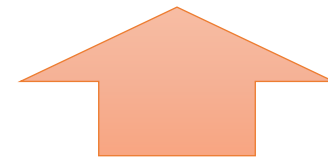


前ページまでの考察のどこにも
分散的な要素は出てこなかった

そもそも論（もう一度）

- ブロックチェーンを何のために使うのか
 - 情報が「改ざんされていないこと」を確認できるようにする
 - 情報が「存在していたこと」を否定できないようにする
 - 上記2つを誰でも実施できるようにする
- これを実現するのに、必ずしも分散システムである必要はない

分散システムにすべきかどうかは、別の要件に依存する



前ページまでの考察のどこにも分散的な要素は出てこなかった

要件の検討

考えるべき要件＝何をしたいのか？

- 情報が「改ざんされていないこと」を確認できるようにする
- 情報が「存在していたこと」を否定できないようにする
- 上記2つをコンソーシアムの参加者なら誰でも実施できるようにする

そもそも論で
検討済み

- 情報を誰が保管するのか？
- 情報が正しくなかったときに被害を受けるor責任を取るべきなのは誰か？

分散システムにすべきかどうか

- 考えるべき項目



- 情報を誰が保管するのか？

- 情報の正しくなかったときに被害を受けるor責任を取るべきなのは誰か？

- 要件：ちゃんといつでも情報を確認できるなら誰か一人が保管していればよい
 - それなら、分散ではなく集中型のシステムで情報を管理すればよい
 - ただし、信頼性・可用性を高めるために、情報は冗長管理しておいたほうがいい
- 要件：情報管理者が業務履行不能になるリスクを考慮しておきたい
 - 情報は複数参加者（場合によっては全参加者）で冗長保管する → 分散システム
- 要件：情報管理者が裏切る（不正を行う）リスクを考慮しておきたい
 - 情報は複数参加者（場合によっては全参加者）で冗長保管する → 分散システム
 - あと、ビジネスとして参加者間でしっかりとした契約（罰則を含む）を締結する

誰が情報を処理すべきか＝誰が署名すべきか（分散システムの要件とは無関係）

- 考えるべき項目

- 情報を誰が保管するのか？



- 情報の正しくなかったときに被害を受けるor責任を取るべきなのは誰か？

- 要件：ある参加者が自身の情報の存在・来歴を証明したい

- その参加者の署名を情報に付けておけば良い

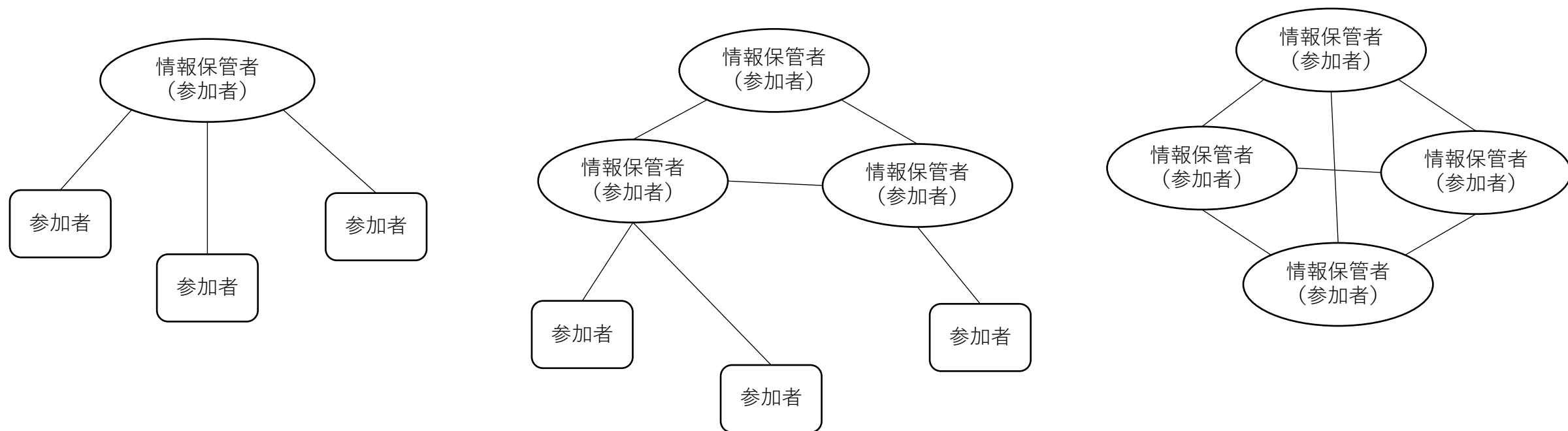
- 要件：複数の参加者間で情報・価値のやり取りを行いたい

- やり取りの当事者全員の署名を情報に付けておけばよい

- 要件：トークンのような総量が決まったものの価値のやり取りを行いたい

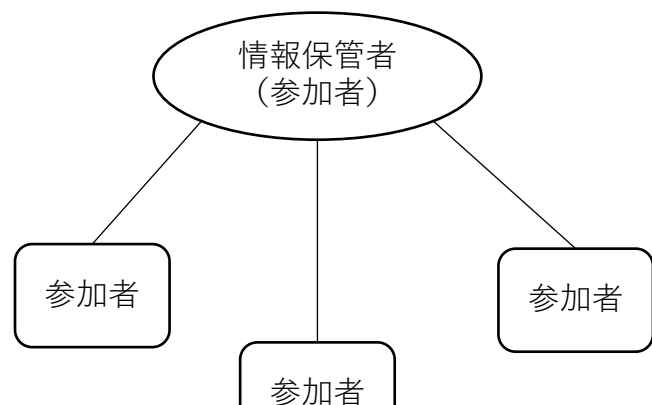
- トークン発行者（不正があったときに被害を受ける人）と、やり取りの当事者の署名を情報に付けておけばよい

システムのパターン

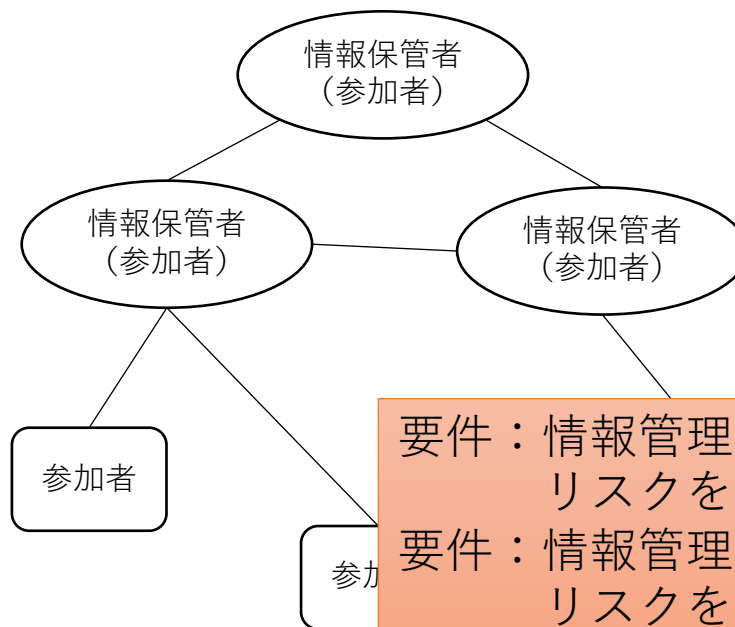


この2つに本質的な違いはない

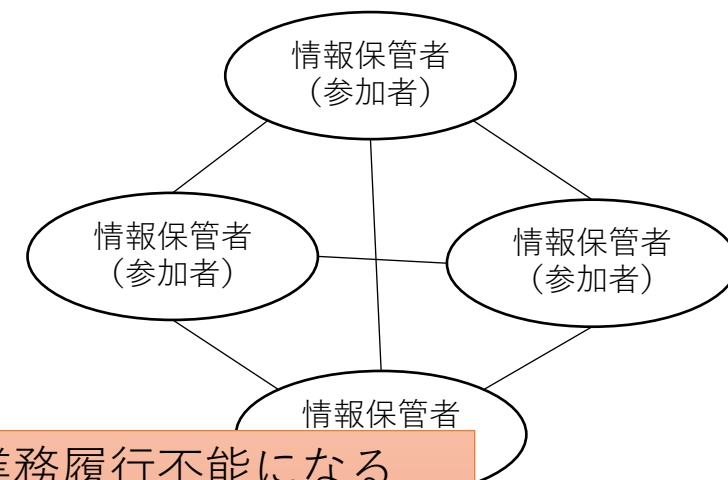
システムのパターン



要件：ちゃんといつでも情報を確認できるなら誰か一人が保管していればよい

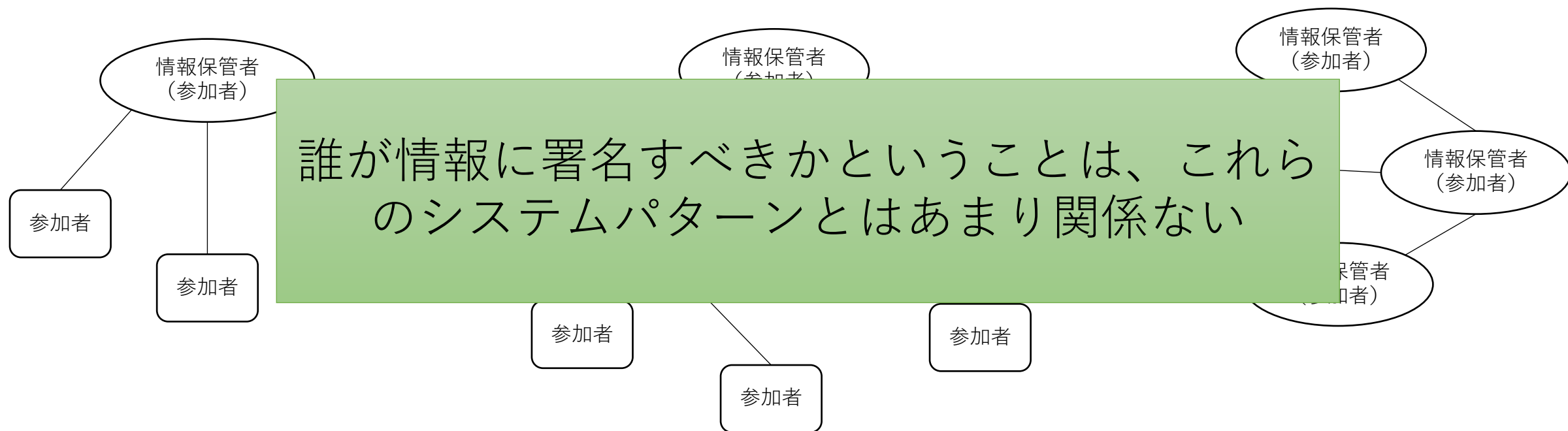


要件：情報管理者が業務履行不能になるリスクを考慮しておきたい
要件：情報管理者が裏切る（不正を行う）リスクを考慮しておきたい



この2つに本質的な違いはない

システムのパターン



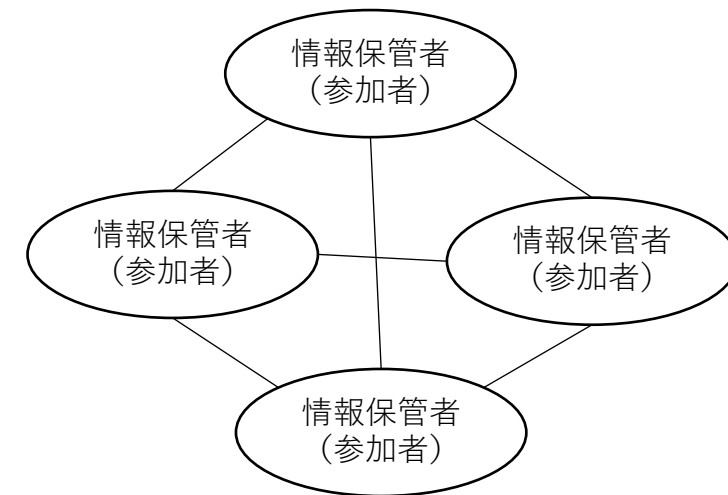
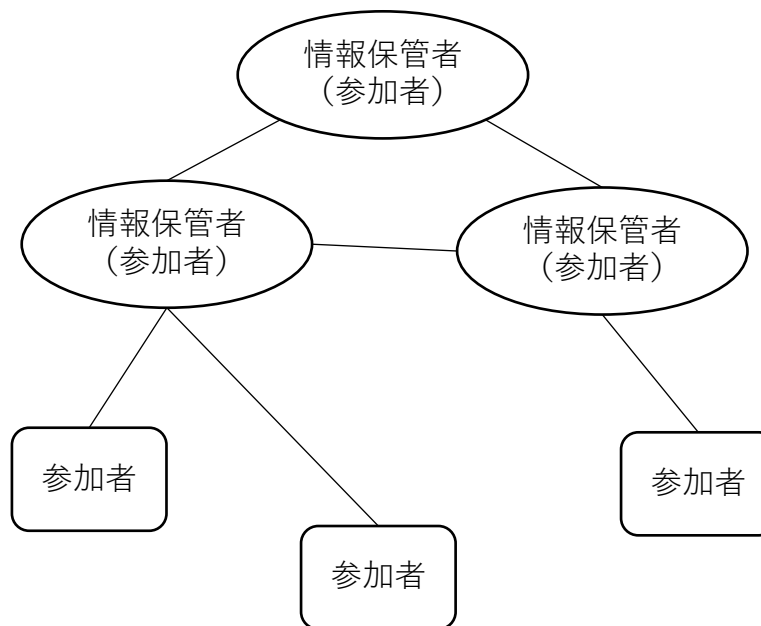
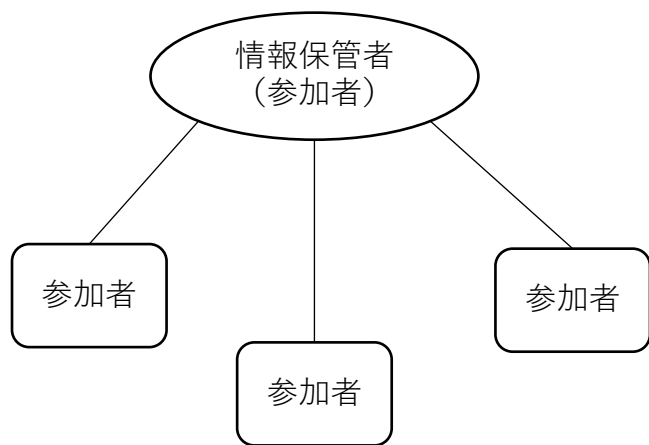
この2つに本質的な違いはない

BBc-1によるシステム化

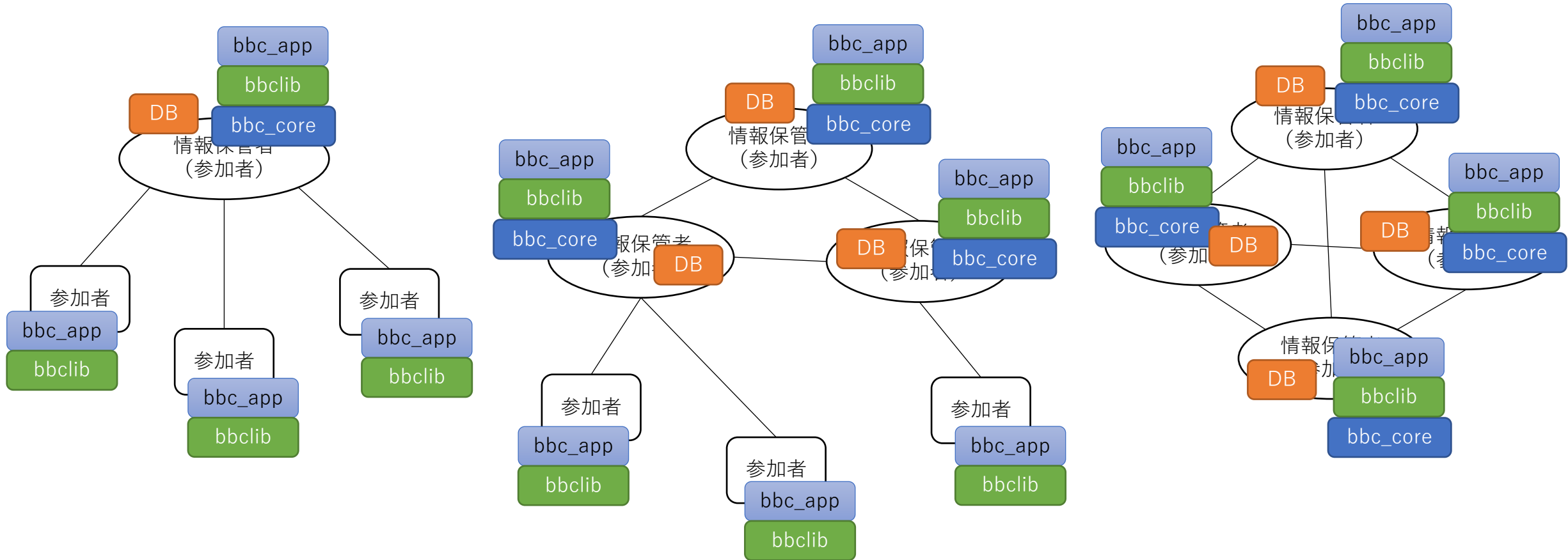
BBc-1の主要な機能・モジュール構成

モジュール（プログラム）名	役割
bbc_core.py	DBへのトランザクション登録、検索の仲介 他のbbc_coreプロセスとのトランザクションの共有 他のユーザとのメッセージング 他のbbc_coreプロセスとのメッセージング
bbclib.py	トランザクションデータ構造の定義
bbc_app.py	bbc_coreとのインタフェース提供（クライアントアプリ側）
database (sqlite3、mysql)	トランザクションデータの保持

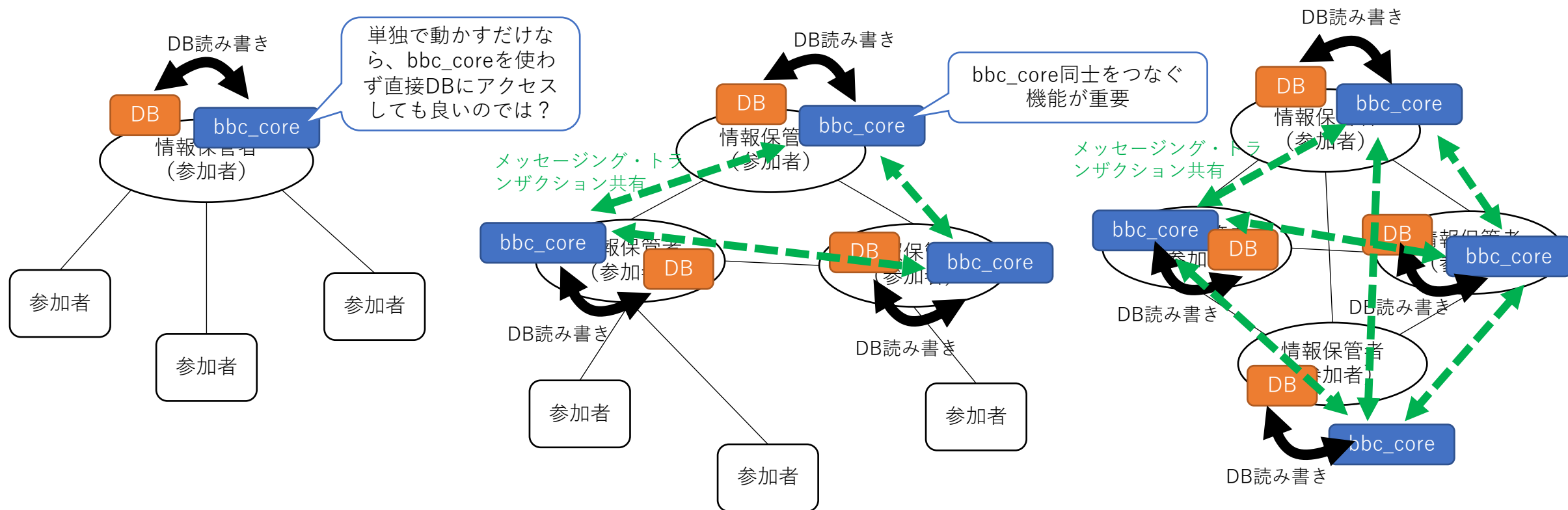
システムパターンとモジュールの対応



システムパターンとモジュールの対応

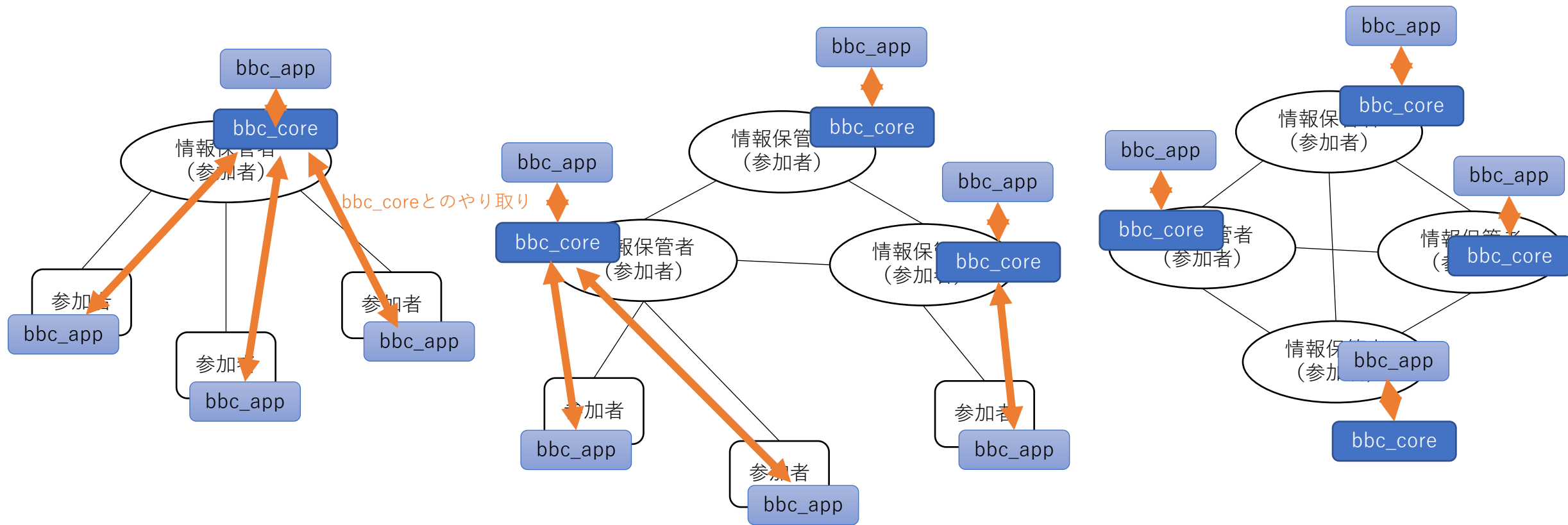


システムパターンとモジュールの対応



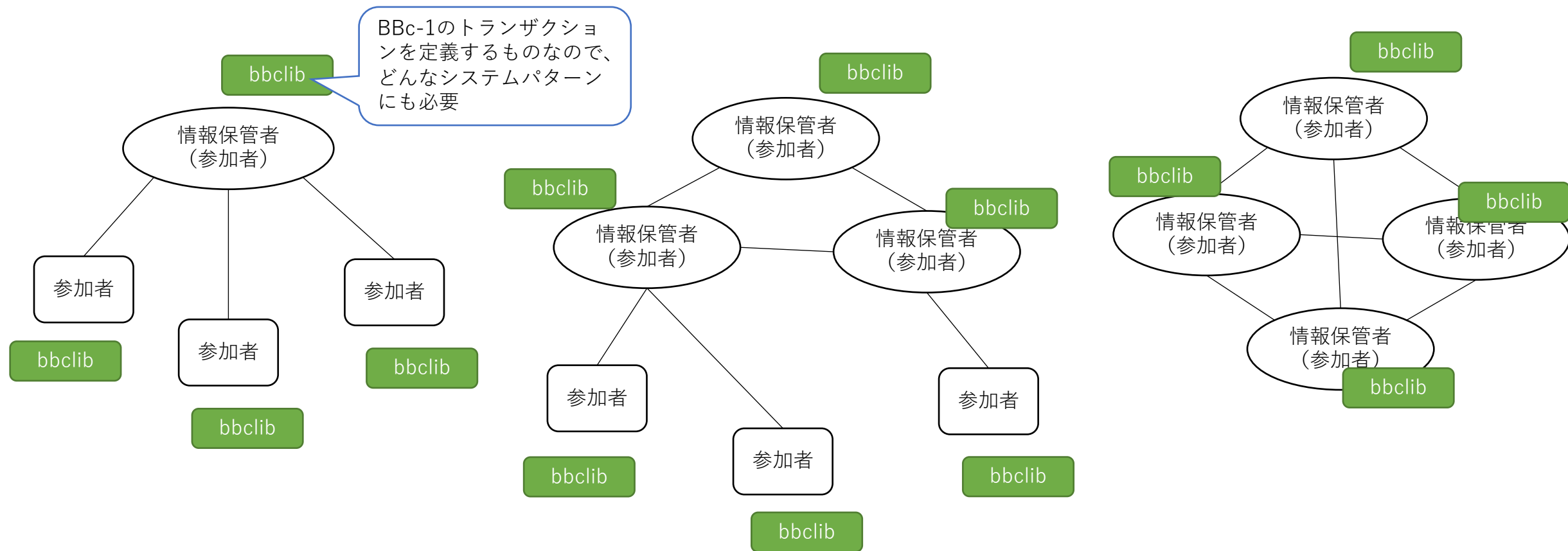
モジュール（プログラム）名	役割
bbc_core.py	DBへのトランザクション登録、検索の仲介 他のbbc_coreプロセスとのトランザクションの共有 他のユーザとのメッセージング 他のbbc_coreプロセスとのメッセージング
database (sqlite3、mysql)	トランザクションデータの保持

システムパターンとモジュールの対応



モジュール (プログラム) 名	役割
bbc_app.py	bbc_coreとのインタフェース提供 (クライアントアプリ側)

システムパターンとモジュールの対応



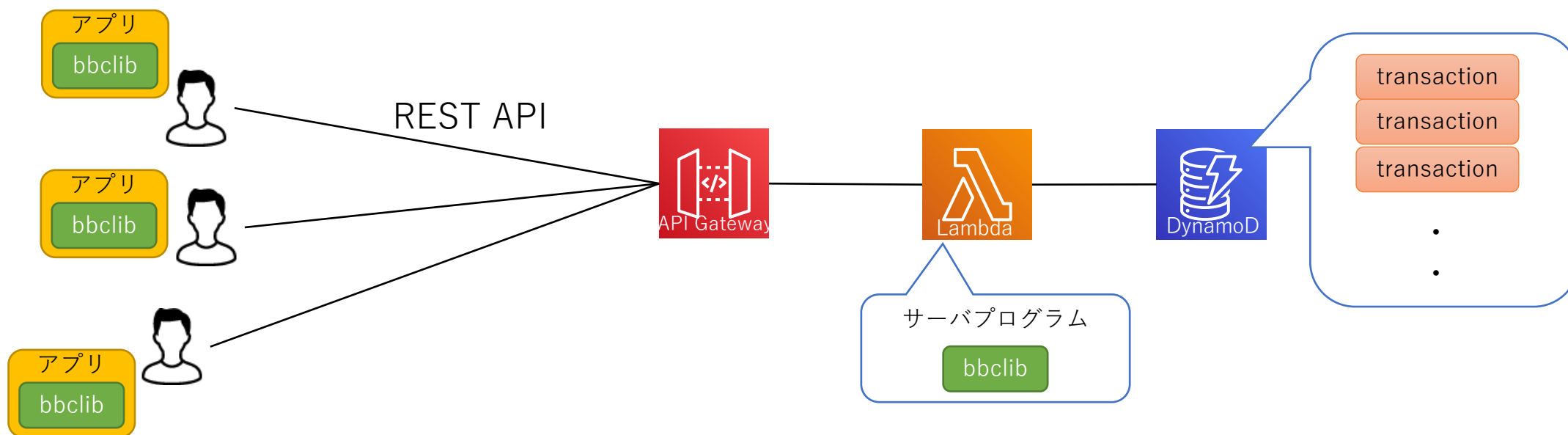
モジュール（プログラム）名	役割
bbclib.py	トランザクションデータ構造の定義

結論

- BBc-1 の概念を利用する場合、最も大切なのはトランザクションとそこに付与される署名である
 - つまり、bbclibはどんなシステムを用いるにしても必要
 - bbclibで生成されたトランザクションデータを参加者が検索可能（削除不能）なDBに格納しておけばよい
 - 逆にいえば、bbclibを用いないシステムはBBc-1ではない
- 中央集中型システムにする場合にはbbc_coreを使う必要はない
 - それに伴い、bbc_appも必要なくなる
 - 例えばクラウドサービスで提供されるサーバレスアーキテクチャを利用する、など
- 細かい要件に応じてbbc_coreと同等の役割を担う別の実装を作っても良い
 - 大事なのはbbclibで生成されたトランザクションを保存したり共有したりできること
 - coreという名前がついているが、中核的な要素ではない

応用事例

- AWS LambdaとDynamoDBを利用したシステム例
 - トランザクションに複数の署名を付与したい場合は、メッセージングの仕組みが必要になる
 - メッセージングのためだけにbbc_coreを利用するのは効率が悪いので、メッセージングの仕組みを別途開発/導入したほうがよいと考えられる
 - 例えば、Lambdaを経由する、Redisなどを利用する、など



以上