COUCHBASE SDK

河野　泰幸 | ソリューション・エンジニア

2021年 5月 12日

アジェンダ

01/ 概要

02/ ブートス トラップ

03/ データ アクセス

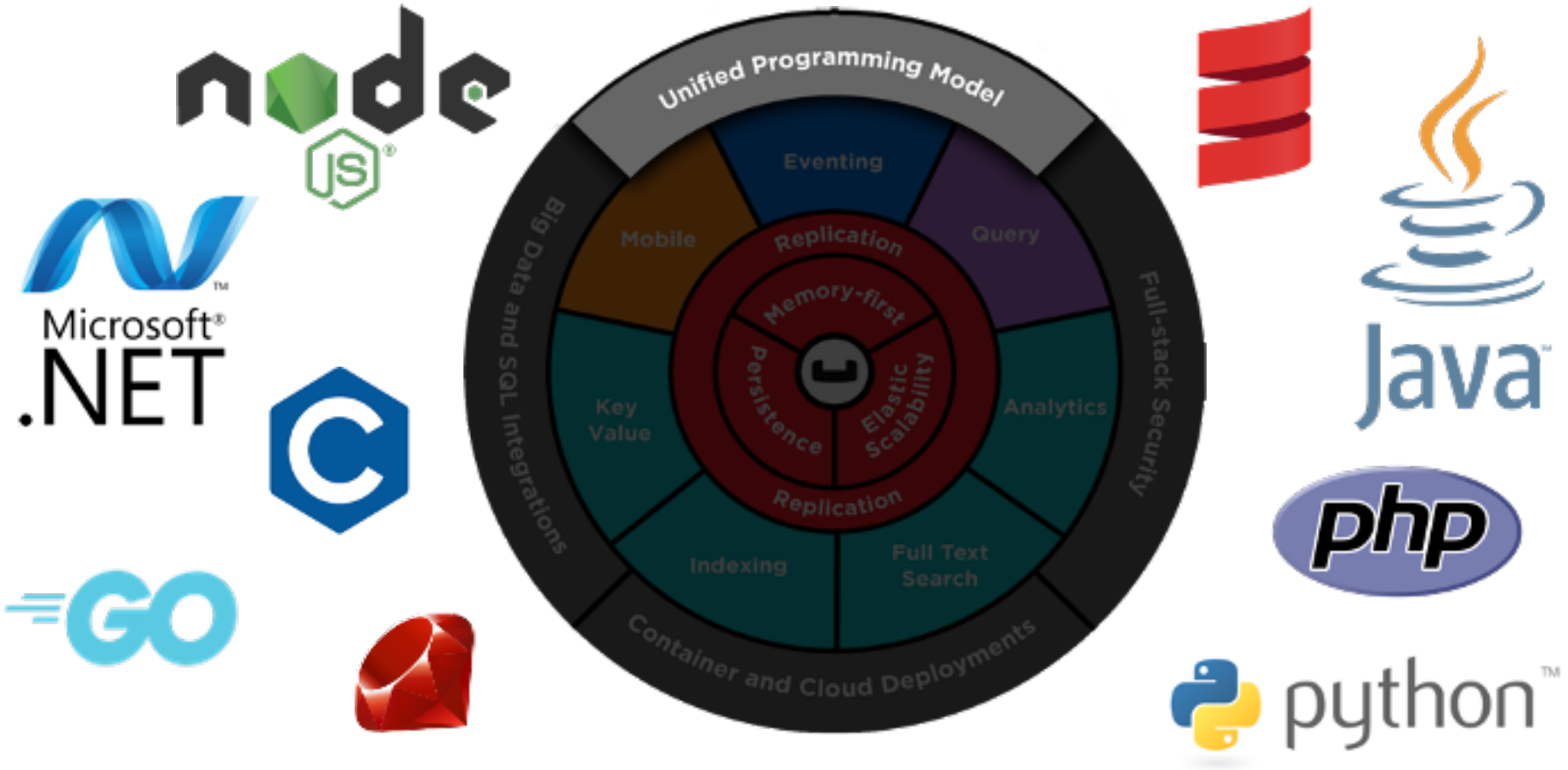
04/ 永続性と一貫性のオプション

05/ エラー処理

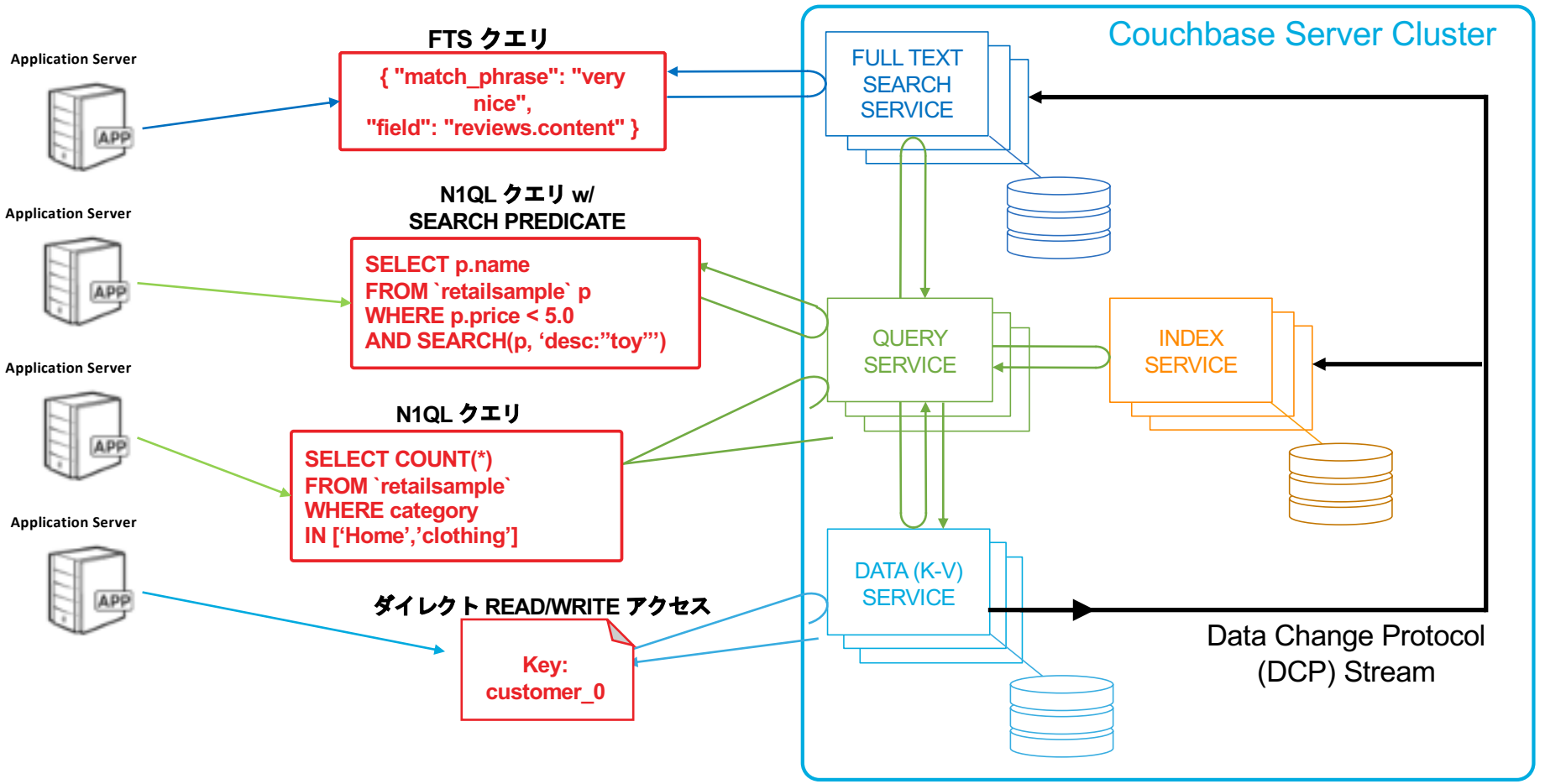
06/ ログとトレース

1 概要

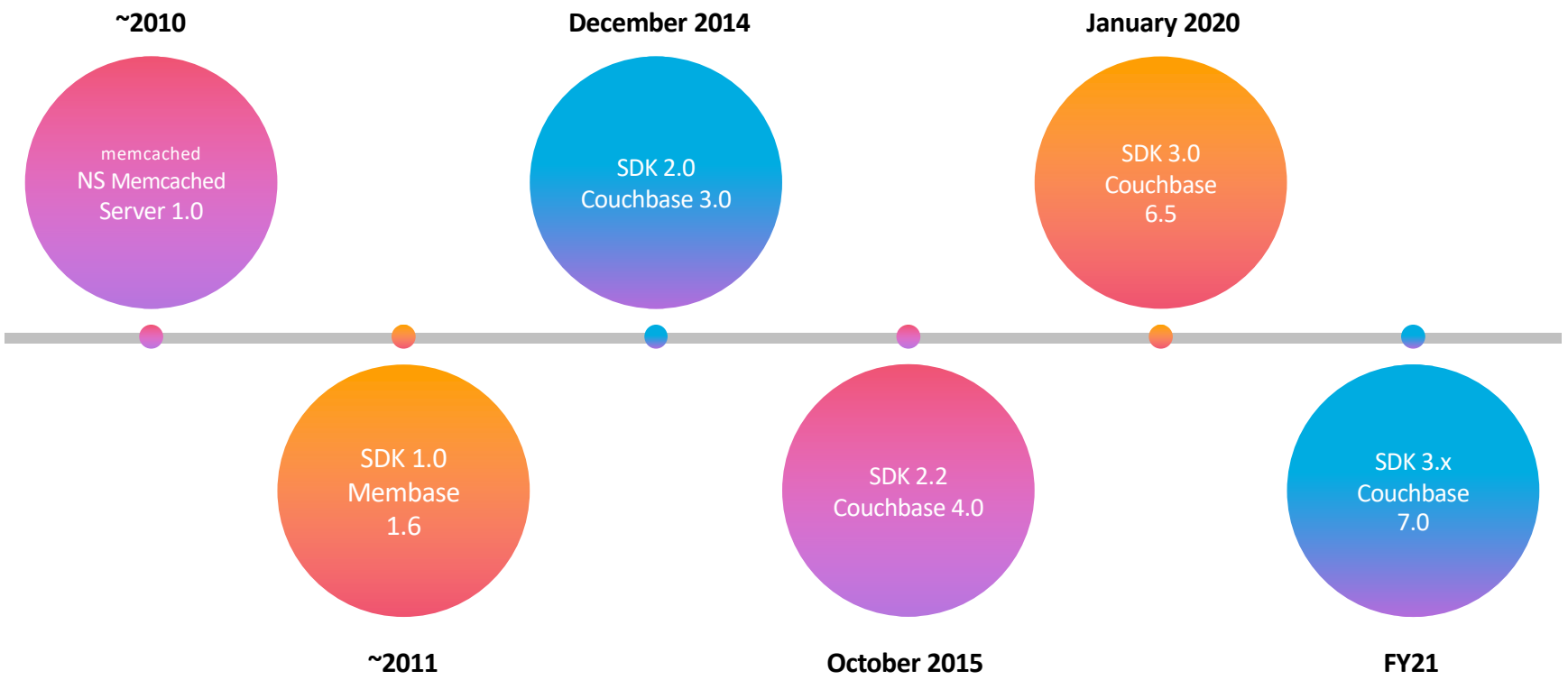
Couchbase Data Platform



アプリケーションとCouchbaseとの対話方法



SDK タイムライン



SDK 3.xの背後にある動機

最新の技術動向に同期

新しい言語バインディング

APIの簡素化と統一

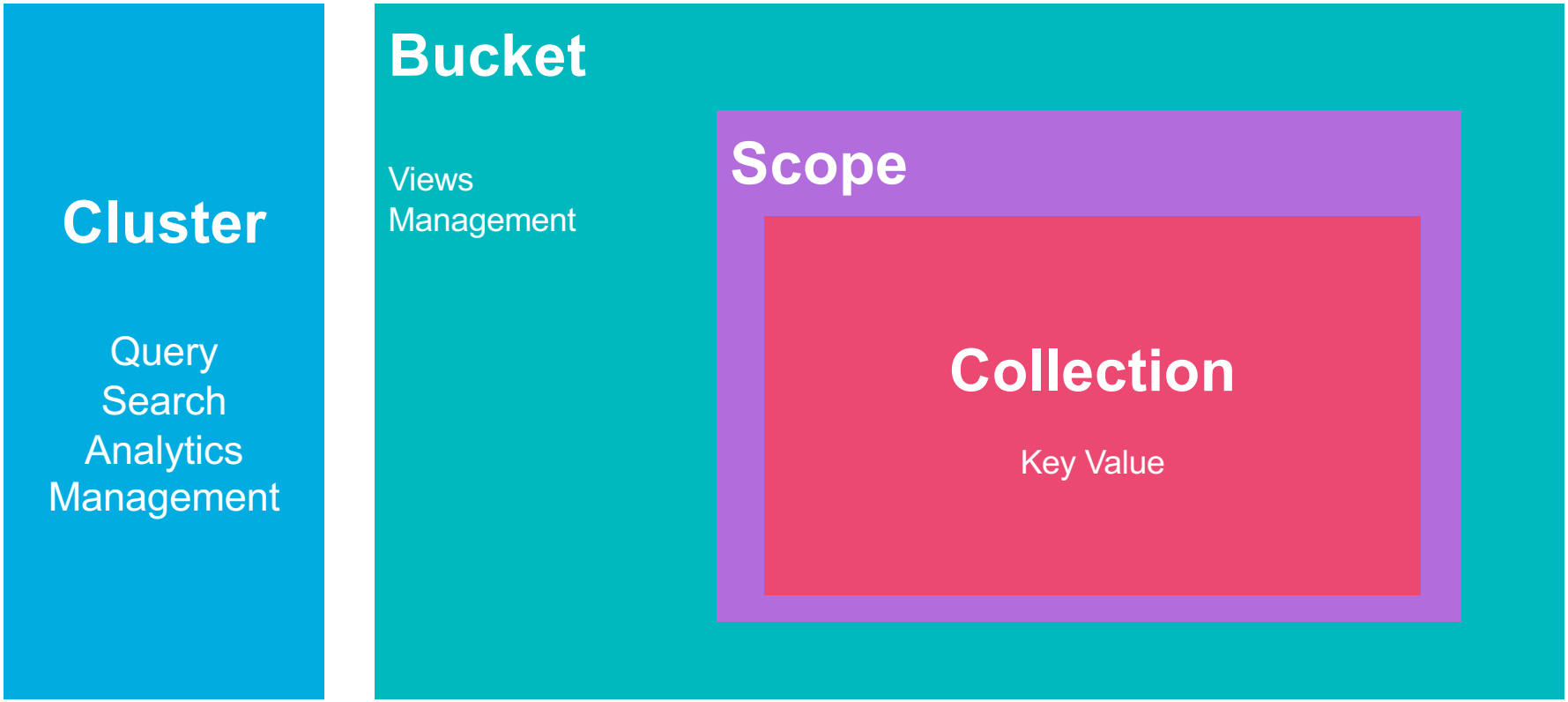
追加のテレメトリ

今後の機能に備える

Collection と Scope

マルチドキュメント ACID トランザクション

各操作がどのコンポーネントに対応しているか



2 ブートストラップ

ブートストラップの基本

```
cluster = Cluster.connect("connection-string", 1
            Authentication{}, 2
            Options{} ); 3

bucket = cluster.bucket("bucket-name"); 4

collection = bucket.defaultCollection(); 5
```

1 - 接続文字列

Formats	Hostname (KV node)
http://<hostname1>,...,<hostnameN>	DNS SRV Record
couchbase://<hostname1>,...,<hostnameN>	FQDN
couchbases://<hostname1>,...,<hostnameN>	IP Address

- 最低必要なホスト名は 1 つ、複数ホスト名指定を推奨
- 指定するホスト名はデータ (KV) ノードである必要があります
- FQDN利用を推奨（IP アドレスより）
- “couchbases://” の最後の「s」は、セキュア接続を意味する。
- DNS SRVレコードの使用（複数のノードをリストしたくない場合）

2 - 認証

- RBAC – ユーザー認証要
- 基本的ルールとして、クラスターに接続するアプリ/サービスごとに 1 つのユーザー アカウントを持つ

3 – 接続オプション

- セキュリティ(i.e. x509)
- 接続設定
- ログ・トレース
- ほとんどの場合、デフォルト構成設定を推奨

4 – バケット

- Couchbase < v6.5: 少なくとも 1 つのオープン バケットが必要
- アプリケーションごとに 1 つのバケット接続のみが必要（シングルトン/マルチトンパターン）
- スコープとコレクションへのアクセスを提供

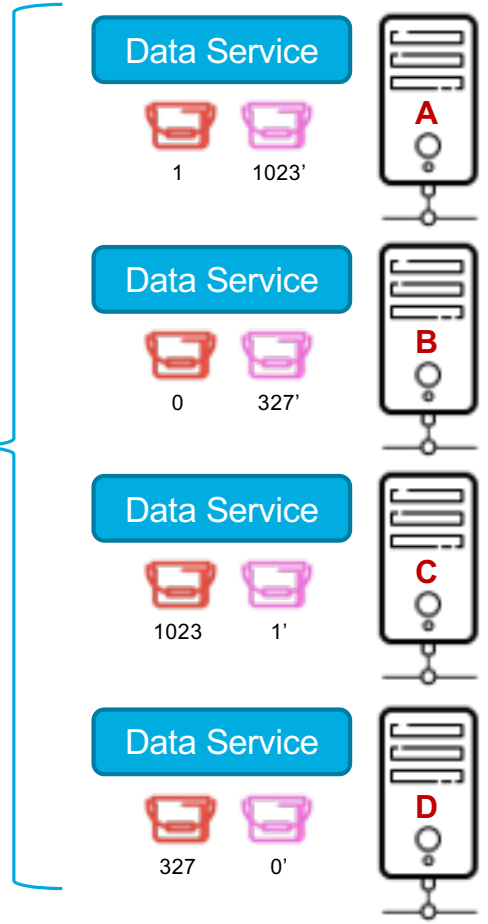
5 – コレクション

- SDK 3.x + Couchbase >= v6.5
- 全てのKV操作のためのプライマリAPI

クラスターマップ

クラスターマップ

vBucket	active	replica 1
0	B	D
1	A	C
…	…	…
327	D	B
…	…	…
1023	C	A



ノート

クラスターマップには、Dataサービス以外のCouchbase サービス (クエリ、FTS など) を実行しているマシンのホスト名や割り当てられたポートなどの情報も含まれています。

ブートストラップのベスト プラクティス

109

Couchbase を使用する場合のコネクション管理のためのベストプラクティスはなんですか?

Couchbase Connect

asked Jan 1 '15 at 10:03
NoSQLNewbie0345
950

Answer

3,005

*** 可能な限り、デフォルトの接続設定を使用することをお勧めします。***

タイムアウト:
ほとんどのアプリケーションでは既定のタイムアウトが機能しますが、可能な範囲で、タイムアウト時間をできるだけ短くしておくことをお勧めします。バッチ操作は完了に時間がかかる場合があるため、特別にタイムアウト時間を考慮する必要があります。

接続:
クライアント SDK は、サービスごとに 1 つの接続を開きます。(K/V, Query, Index, etc.). つまり、K/V を除くすべてのサービスは、必要に応じて動的に拡張される接続プールを持ちます。 非常に大きなドキュメントの送受信など、特定のユースケースでのみ、これらの接続のデフォルトを変更する必要があります。そして、可能な場合は接続を閉じます。

キープアライブ:
クライアント アプリと Couchbase クラスターの間には、しばしばファイアウォールが配置される事があるため、キープアライブ間隔とタイムアウトにご注意ください。

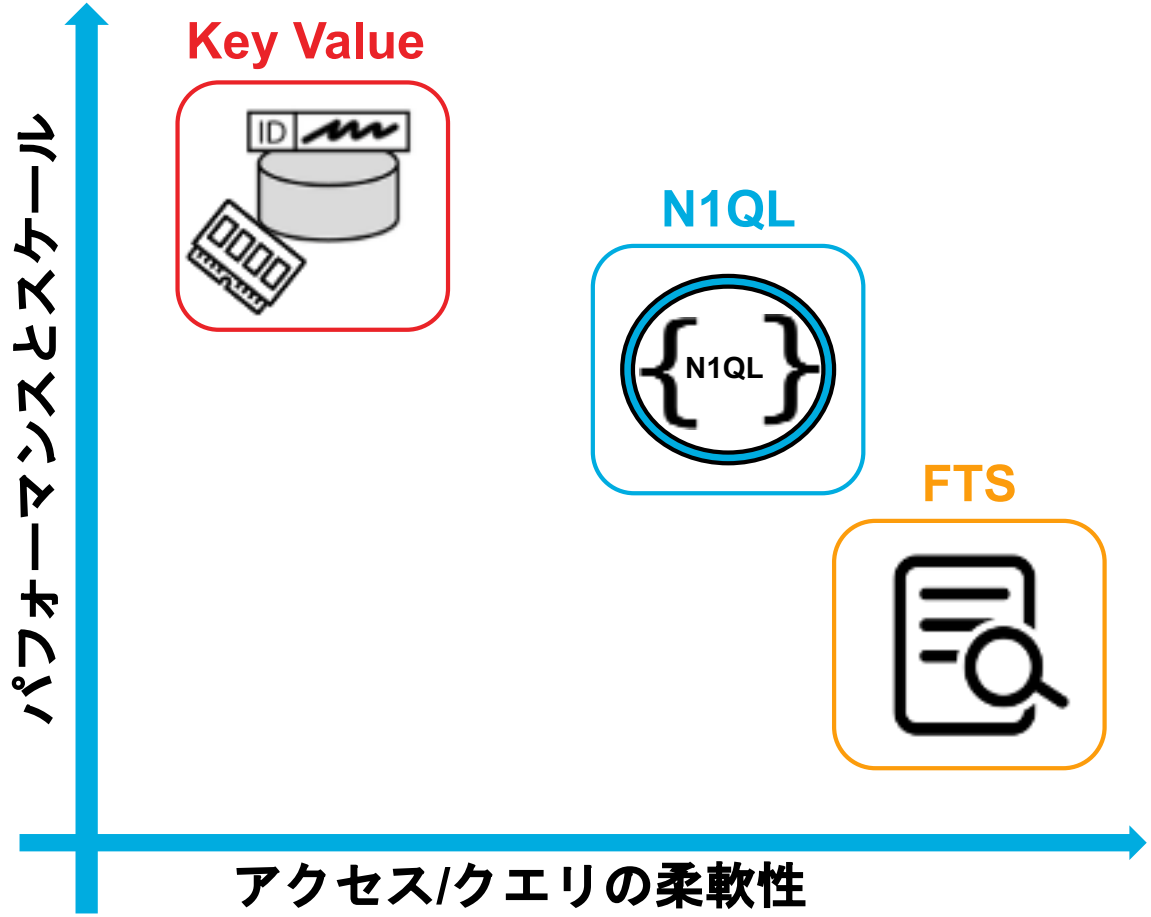
answered Jan 1 '15 at 10:03
CouchbaseGuru1
950k

常に徹底的に設定をテストしてください! SLA が満たされていることを判断するには、それ以上に良い方法はありません。

-- CouchbaseUser6.5 Jan 1 '15 at 10:15

3 データ アクセス

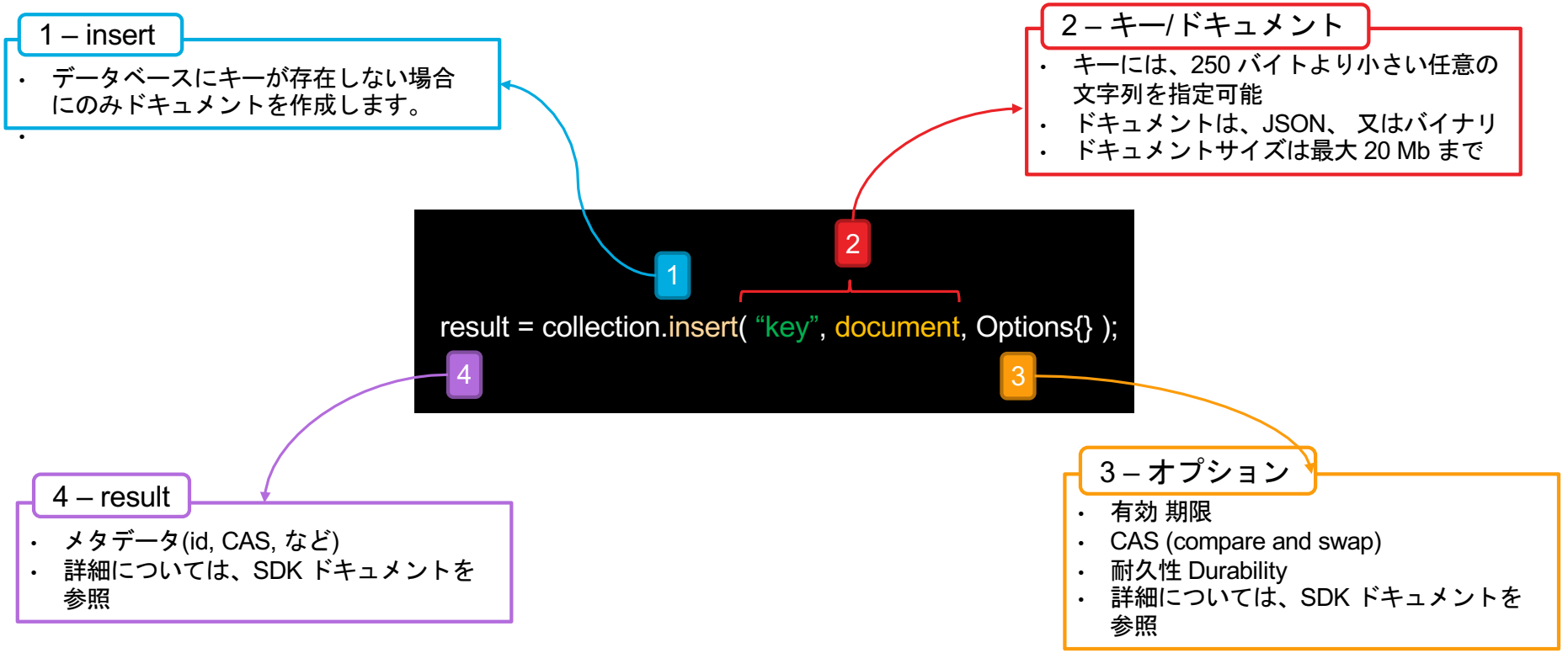
仕事に適した道具を使用する



利用可能な Couchbase データ アクセス方法

Couchbase API	レイテンシ	スループット	スケーラビリティ	適用性
Key-Value	50µs – 10 ms	> 10M ops/sec	非常に高	高スループット、レイテンシに敏感なワークロードに最適
N1QL	< 1ms - 2ms+	> 40k query/sec	高	セカンダリ ルックアップ、データベースにロジックをプッシュする場合に最適
Full Text Search	5ms+	> 20k query/sec	高	自然言語テキスト検索、関連性ランキング、ファセット化
Analytics	1s – 60s+	< 1-10 query/sec	高	非常に大きなデータ量の複雑な分析またはアドホック分析

K/V – CRUD オペレーション



K/V – CRUD Operations

1 – get

- データベースにキーが存在する場合にドキュメントを取得します。
-

2 – キー

- 取得するドキュメントのキー/ID

4 – result

- メタデータ(id, CAS, etc.)
- ドキュメント値
- 詳細については、SDK ドキュメントを参照

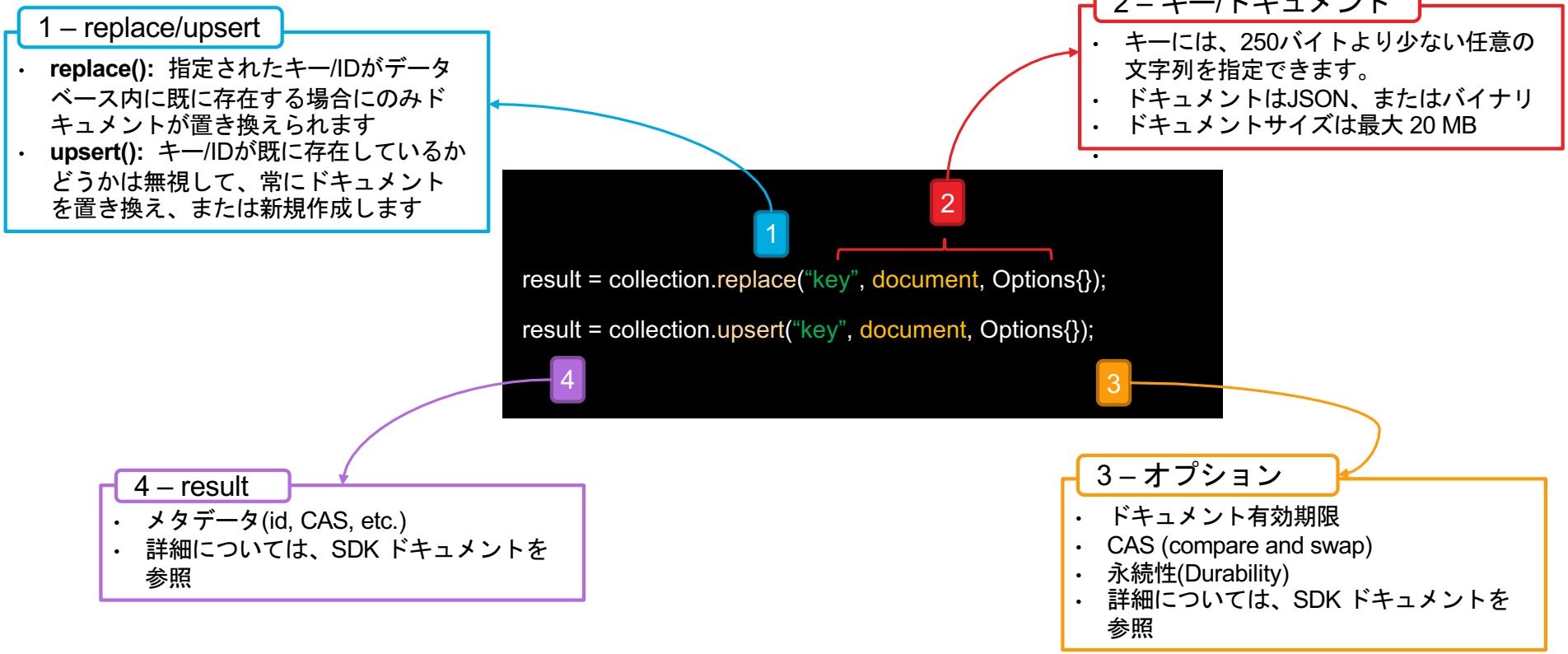
3 – オプション

- タイムアウト
- 詳細については、SDK ドキュメントを参照

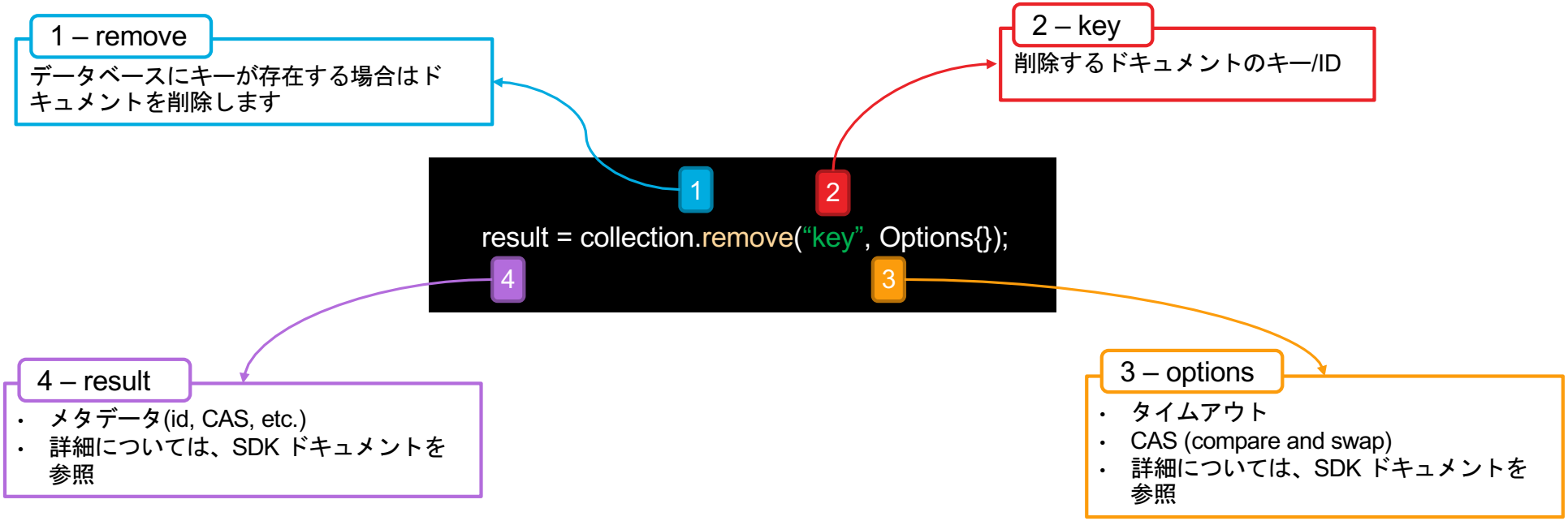
同様の操作

- **getAnyReplica("key", Options{}):** レプリカノードからドキュメントを取得する
- **getAndLock("key", time):** ドキュメントを取得し、指定された時間(秒単位)でロックします(悲観的ロック) 。unlock() メソッドを使用してドキュメントのロックを解除する

K/V – CRUD Operations



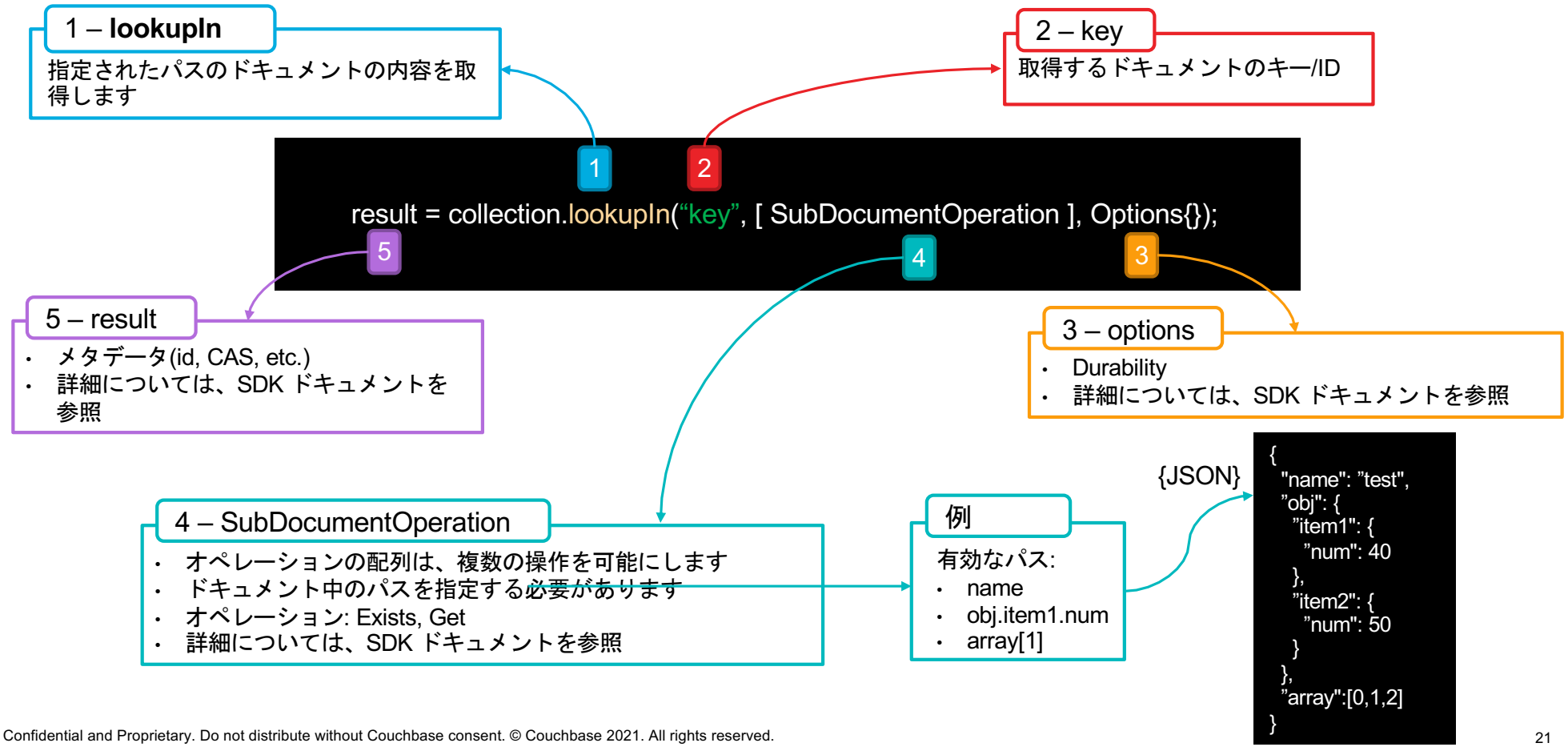
K/V – CRUD Operations



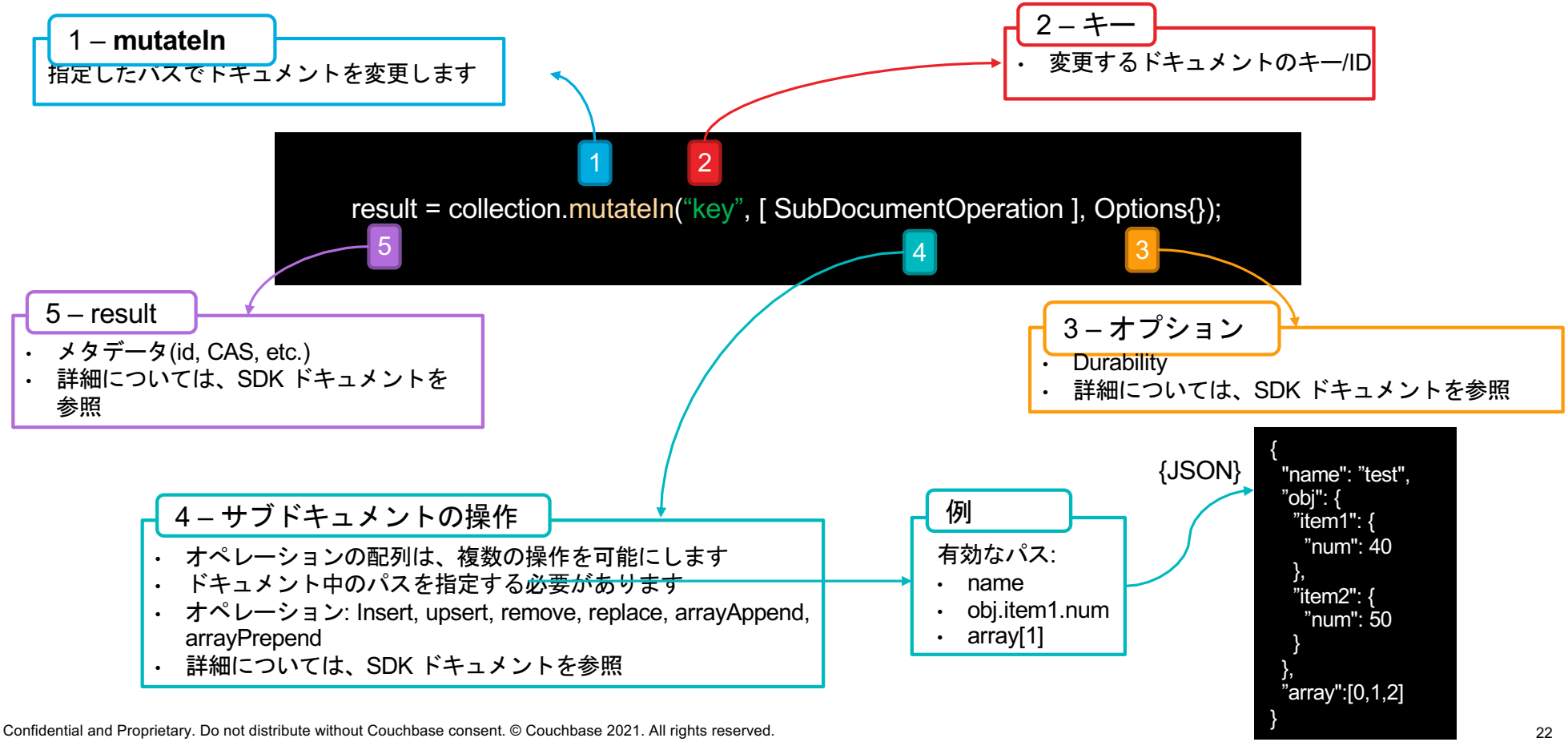
Similar operations

- **touch("key",time):** 指定されたキー/IDのドキュメントの有効期限をリセットします
- **getAndTouch("key", time):** 指定されたキー/IDのドキュメントを取得し、有効期限をリセットします

K/V – サブドキュメント操作– lookupIn()



K/V – サブドキュメント操作– mutateIn()



バッチ/非同期操作

39

K/V操作をバルクで行うことは可能ですか?

Couchbase K/V Batch Ops

asked Jan 1 '15 at 10:03

user1234567
10

Answer

5,030

バッチ操作:

ほとんどの SDK には、マルチ K/V API が用意されています。通常、マルチ API は、Multi が追加された K/V 操作です (つまり、get() K/V 操作に対する、 getMulti() バッチ API)。マルチ API の詳細については、各 SDK を参照してください。

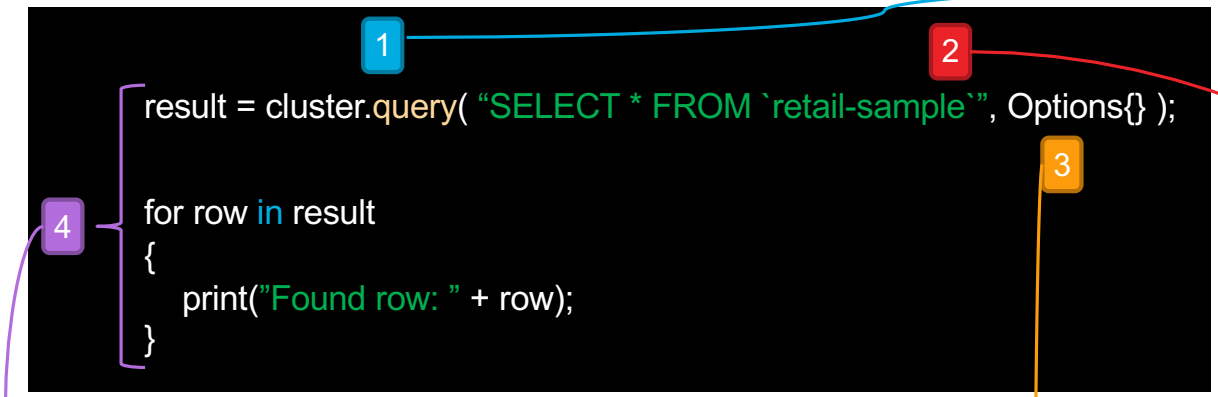
非同期操作:

3.x SDK 以降では、バッチ処理を処理するために非同期操作を利用可能であり、推奨されます。
各 3.x SDK には、パフォーマンスを向上させるためにバッチ操作を使用するために使用する非同期 API が用意されています。

answered Jan 1 '15 at 10:03

CouchbaseGuru1
950k

N1QL - 基本



1 – クエリ

- SDK < 3.x: クエリはバケットレベル
- Couchbase < 6.5: 少なくとも 1 つのバケットが開いている必要があります
- SDK >= 3.x: クエリはクラスタ レベル

2 – クエリ文字列

- N1QL ステートメントの文字列表現
- 構文と詳細については、N1QL のドキュメントを参照ください。

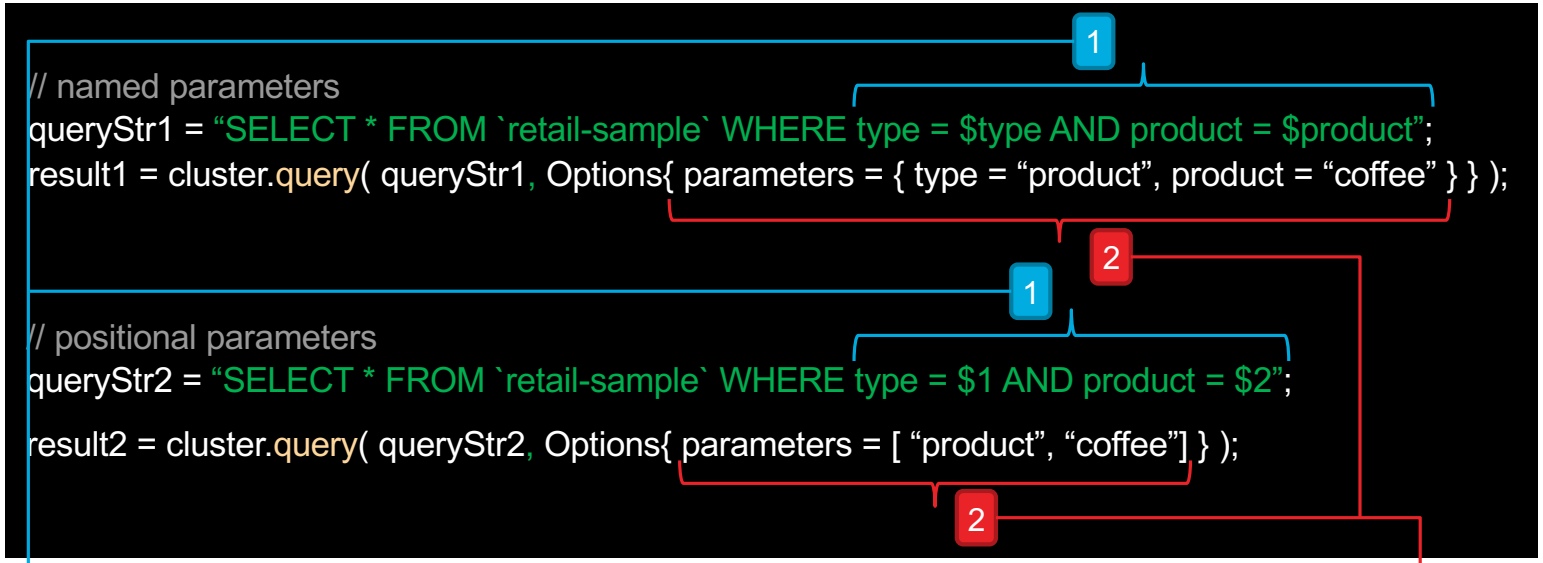
3 – オプション

- クエリ パラメーター
- プリペアドクエリ
- スキャンの整合性
- 詳細については、SDK ドキュメントを参照

5 – result

- N1QL 実行から返されたドキュメントのリスト
- 詳細については、SDK ドキュメントを参照

N1QL - パラメーター化



重要

名前付きパラメーターまたは位置指定パラメーターは、クエリの WHERE 節、LIMIT 節、または OFFSET 節の値に対して使用されます。

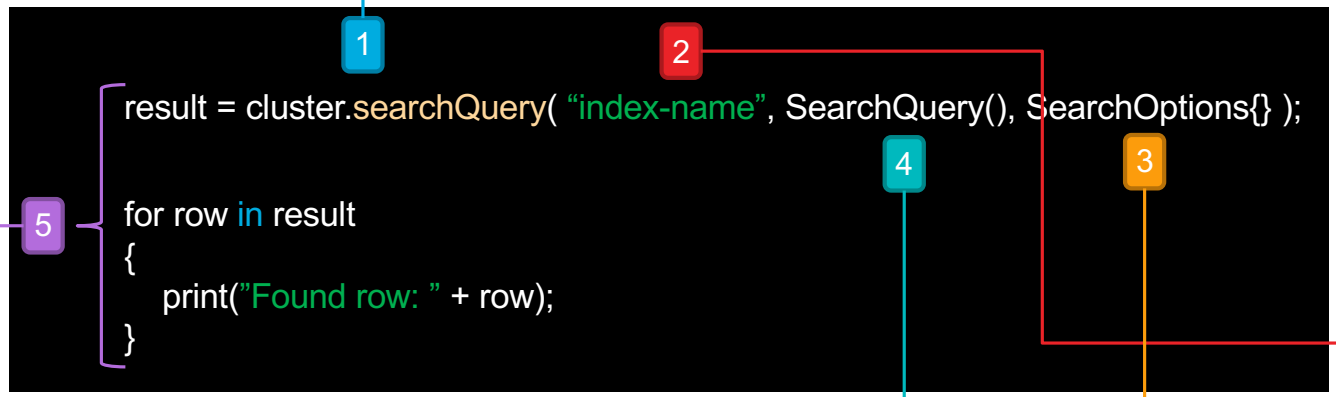
1 – クエリ文字列

- N1QL ステートメントの文字列表現
- 名前付きパラメーターは $<param name> 構文を使用
- 位置パラメータは$1、$2などの構文を使用

2 – パラメーター

- 名前付きパラメーターは、キーが名前付きパラメーターと同じ名前を持ち、値が名前付きパラメーターの値を保持するキーと値のペアです。
- 位置パラメータは、クエリ内の指定された位置を介して渡されます。 したがって、クエリ中の $1 は最初のパラメータ $2は2 番目のパラメータ（以下同様）を参照します。
- 詳細については、SDK ドキュメントを参照

Full Text Search



1 – searchQuery

- SDK < 3.x: searchQueryはバケットレベル
- Couchbase < 6.5: 少なくとも 1 つのバケットが開いている必要があります
- SDK >= 3.x: searchQueryはクラスタ レベル

2 – インデックス名

- クエリで使用するフルテキスト検索インデックスの名前

3 – 検索オプション

- 制限とスキップ
- ハイライト
- スキャンの整合性
- 詳細については、SDK ドキュメントを参照

5 – result

- FTS クエリから返される結果のリスト
- スコア、フラグメントなどが含まれています。
- 詳細については、SDK ドキュメントを参照

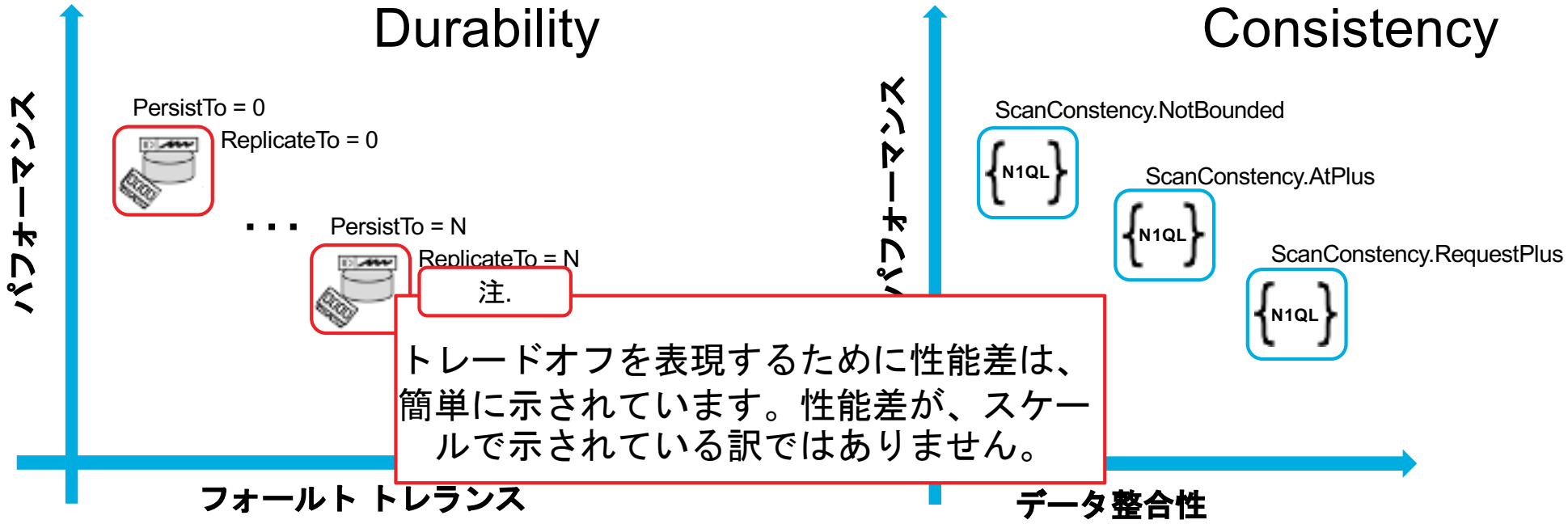
5 – SearchQuery

- Simple, Match, MatchPhrase, など.
- Ranges
- Conjunction(AND)と Disjunction(OR)
- 詳細については、SDK ドキュメントを参照

4

永続性と一貫性のオプション
DURABILITY & CONSISTENCY

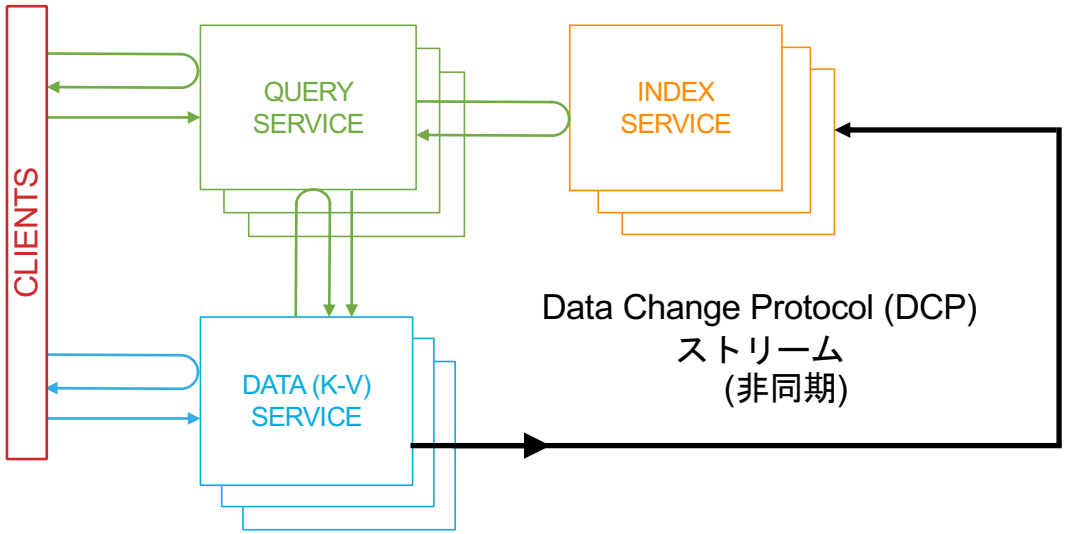
永続性 (Durability) vs. 一貫性 (Consistency)



- キー/バリュー操作
- 障害に対する保護
- レプリケーションの程度や永続性を選択可能
- デフォルトでは永続化は非同期 (レプリカへの反映とディスクへの永続化を待たず、メモリレベルで更新処理が完了)

- N1QL操作
- データとインデックスとの間の同期
- Choose degree of replication and/or persistence
- デフォルトは「結果整合」(インデックス更新は非同期処理)

一貫性 - Read your own writes (RYOW)



RYOW

- インデックスは非同期的に更新されます。
- そのため、インデックスの内容は、データよりも古い(遅れている) 可能性があります。
- **MutationTokens** – クラスターの環境でMutationTokenを有効にする必要があるかどうかの詳細については、SDKをチェックしてください。トークンを使用すると、メタデータが追加されるためにネットワークにいくらかのオーバーヘッドが発生します
- **ScanConsistency.REQUEST_PLUS** - 別の、より厳しいオプション

```
written = collection.upsert( “key”, document, Options{} );

queryStr = “SELECT * FROM `retail-sample`”;

result = cluster.query( queryStr, Options{ ConsistentWith: written } );
```

Scan Consistency

次のオプションを使用できます。

•not_bounded：クエリの一貫性を必要とせずに、クエリをすぐに実行します。インデックスのメンテナンスが遅れている場合は、古い結果が返されることがあります。

•at_plus：最初にインデックスを最後の更新のタイムスタンプに更新されている事を保証して、クエリを実行します。index-maintenanceが遅れている場合、クエリは追いつくのを待ちます。

•request_plus：最初にインデックスを現在のクエリリクエストのタイムスタンプに更新されている事を保証して、クエリを実行します。index-maintenanceが遅れている場合、クエリは追いつくのを待ちます。

N1QLの場合、デフォルトの整合性はnot_boundedです。

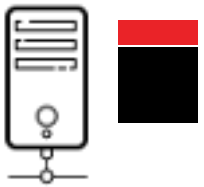
https://docs.couchbase.com/java-sdk/current/concept-docs/n1ql-query.html

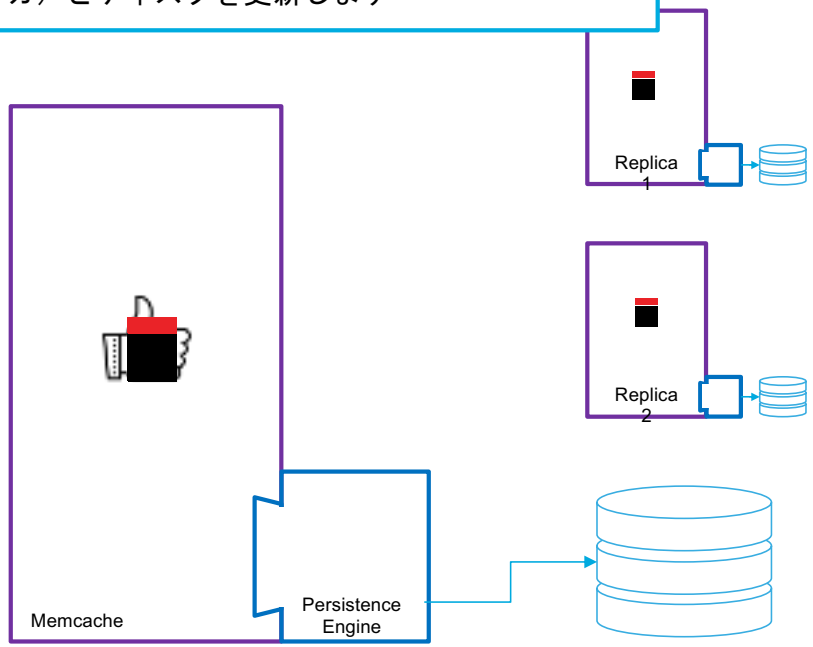
同期レプリケーション(Couchbase Server 6.5 + SDK 3.x)

永続性レベル

- **None** – デフォルトの、結果整合性と永続化モデル
- **Majority** – リターンする前に、過半数のコピーのメモリ（レプリカ）を更新します
- **MajorityAndPersistToActive** – リターンする前に、過半数のコピーのメモリ（レプリカ）とアクティブなvBucketのディスクを更新します
- **PersistToMajority** – リターンする前に、過半数のコピーでメモリ（レプリカ）とディスクを更新します

```
result = collection.upsert("key", document,
            Options{ DurabilityLevel.MAJORITY } );
```





ACID トランザクション (Couchbase Server 6.5 + SDK 3.x - Java)

A	Atomicity 原子性	異なるノード上の複数のシャードで複数のドキュメントを更新する際の、オール・オア・ナッシングの更新を保証します。
C	Consistency 一貫性	**レプリカ** はトランザクションコミットとすぐに一致します。 **インデックスと XDCR**は最終的にトランザクションコミットと一致する *(N1QL は、読み取り時の強力な一貫性をrequest_plusオプションにより実現可能)*
I	Isolation 分離性	**「READ COMMITTED」** 同時実行トランザクションのための分離 多くのデータベース(Oracle、PostgreSQL、SQL Server)でデフォルトのトランザクション分離レベル
D	Durability 耐久性	**障害発生時のデータ保護**: 3つの異なるレベル *- ノードの過半数にレプリケートする* *- 過半数にレプリケートし、プライマリのディスクに永続化する* *- ノードの過半数でディスクに永続化する.*

注.

- 同期レプリケーションは、Couchbase で分散された複数ドキュメントの ACID トランザクションをサポートするための基盤となるメカニズムです。
- ドキュメントのコミットされていないバージョンは、k-v メタデータの xattrs セクションに格納されます。これらのバージョンのデータは、Couchbaseのドキュメントが持つ値のサイズに関する 20 MB の制限に含まれます。

トランザクション分離レベルについてのまとめ

https://qiita.com/song_ss/items/38e514b05e9dabae3bdb

ダーティリード (Dirty Read)
トランザクションBでコミットされていないデータをトランザクションAで読み取ってしまう問題

ファジーリード/ノンリピータブルリード (Fuzzy Read / Non-Repeatable Read)
トランザクションAでデータを複数回読み取っている途中で、トランザクションBでデータを更新してコミットした場合、トランザクションAで違う結果のデータを読み取ってしまう問題(非再現リードとも呼ぶ)

ファントムリード (Phantom Read)
トランザクションAで一定の範囲のレコードに対して処理を行っている途中で、トランザクションBでデータを追加・削除してコミットした場合、トランザクションAにデータが反映されるため、処理の結果が変わってしまう問題

- READ UNCOMMITTED：コミットされていない変更を他のトランザクションから参照できる
  - **ダーティリード**、**ファジーリード**、**ファントムリード**が全て発生する
- READ COMMITTED：コミットされた変更を他のトランザクションから参照できる
  - Oracle、PostgreSQL、SQL Serverのデフォルトのトランザクション分離レベル
  - **ファジーリード**、**ファントムリード**が発生する
- REPEATABLE READ：コミットされた追加・削除を他のトランザクションから参照できる
  - MySQLのデフォルトのトランザクション分離レベル
  - **ファントムリード**が発生する
  - MySQL(InnoDB)はREPEATABLE READでも**ファントムリード**が発生しない
- SERIALIZABLE：強制的にトランザクションを順序付けて処理する(直列化)
  - 読み取るすべての行に共有ロックをかける
  - **ダーティリード**、**ファジーリード**、**ファントムリード**が全て発生しない

SDK 3.x (Java, .NET, C++) ACID トランザクション

1 – トランザクション開始

2 – 処理実行

3 – コミット/ロールバック

```
transactions.run((txnctx) -> {
        // Insert a document:
    JsonObject doc1 = JsonObject.create().put("content", "initial");
    txnctx.insert(collection, "doc1", doc1);

    // Replace a document:
    TransactionJsonDocument doc2 = txnctx.getOrError(collection, "doc2");
    JsonObject content = doc2.contentAs(JsonObject.class);
    content.put("name", "bob");
    txnctx.replace(doc2, content);

    // Commit transaction - if omitted will be automatically committed
    txnctx.commit();
});
```

5 エラー処理

エラー処理

```
try {
    result = collection.insert("key", document, Options{});
}
catch (DocumentAlreadyExists){
            …
}
catch (CouchbaseExpection) {
            …
}
```

Exceptions/Errors

- 通常、基底例外クラス(CouchbaseException)を継承したより特定の例外/エラーが用いられる
- ほとんどの操作をエラー処理ロジックでラップすることをお勧めします。
- 回復不可能なCouchbase例外/エラーvs. 言語フレームワークのその他の例外/エラーの一部
- 例外/エラーの種類の構文と詳細については、SDK のドキュメントを参照ください。

考慮事項

- さまざまな種類の例外/エラ毎のアプリケーション ロジック (エラーは一時的なエラーか?等)
  - データ エラー: ドキュメントが存在しない、CAS の不一致など。
  - 一時的/リソースエラー: 一時的な障害、メモリ不足
  - バケット操作エラー: ドキュメントが既に存在する、要求が大きすぎるなど
- 例外/エラーの種類の詳細については、SDK のドキュメントを参照ください。

再試行ロジック

```
MAX_RETRIES = 5
BASE_DELAY = 50

attempt = 1
do {
    try {
        result = collection.upsert("key", document, Options{});
        break;
    } catch (Exception) {
        sleep(BASE_DELAY * attempt);
    }
} while (++attempt != MAX_RETRIES)
```

再試行が意味をなす場合

- 一時的なエラー
- アプリケーションのロジックパスに対して合理的
- 例: durability, query execution exception, index not ready, etc.

再試行が意味をなさない場合

- 非一時的なエラー
- アプリケーションは再試行を必要としないか、失敗条件は非合理的
- 例: bucket doesn't exist, authentication, etc.

その他の考慮事項

- 例外/エラー発生時のフォールバック
- 失敗を想定し、例外/エラーを親に伝播する

再試行に関する考慮事項

- すぐに再試行
- 遅延再試行 (固定, 線形増加, 指数関数的に増加, ランダム)
- **常に**再試行回数最大値を設定する

6 ロギングとトレース

SDK でのログ記録

- なぜログを記録するのか?
  - ログを使用してシステムに関する情報を提供する
  - サポートと協力して作業する場合、DEBUG レベルでログを提供
  - アプリケーションで利用していない場合でも、ログを設定することをお勧めします (問題発生に備える)
- ログレベルとパフォーマンスのトレードオフに注意
  - DEBUG ログは通常は必要ではなく、パフォーマンスが低下します。必要な場合のみ使用してください

SDK を使用したトレース

```
tracer = ThresholdTracer{
        logInterval: 10s,
        kvThreshold: 500ms
};

cluster = Cluster.connect( “connection-string”,
                Authentication{},
                Options { Tracer: tracer } );
```

トレース とは…

- Response Time Observability （応答時間の観測可能性） (Couchbase 方言)
- 閾値ロギング

トレーサー プロパティの例

プロパティ名	デフォルト値
logInterval	10 s
kvThreshold	500 ms
queryThreshold	1 s
sampleSize	10

Note: すべてのトレーサー プロパティの詳細については、SDK のドキュメントを参照ください。

なぜROT(応答時間観測可能性)が必要か?

- 分散システムには、多くの「歯車」が存在しますが、一体どの要素がシステム遅延を引き起こしているのかを特定する必要があります。
- 閾値を設け、応答時間がその閾値を超えていないか監視する方法を提供

SDK を使用したトレース

```
{
  "decode_us": 1203,
  ”last_dispatch_us": 947,
  ”last_local_address": "127.0.0.1:55011",
  ”last_local_id": "41837B87B9B1C5D1/000000004746B9AA",
  ”last_operation_id": "get:0x1",
  ”last_remote_address": "127.0.0.1:11210",
  "server_us": 23,
  "total_us": 1525
}
```

Operation Output

出力フィールド	説明
decode_us	応答をデコードする時間 (マイクロ秒単位)
last_dispatch_us	クライアントが要求を送信し、応答を受け取った時間 (マイクロ秒単位)
last_local_address	操作に使用されたローカルソケット
last_local_id	CBS >= 5.5: サーバーとのネゴシエートの単位として、ログ情報と関連付けるために使用されるID
last_operation_id	操作とIDの組み合わせ、local_idとの組み合わせで診断およびトラブルシューティングに役立ちます
last_remote_address	操作に使用されたサーバー上のリモートソケットのアドレス
server_us	サーバー 側での処理実行に要した時間 (マイクロ秒単位)
total_us	オペレーション実行合計時間

Note

しきい値ログの使用とJaegerのような他のトレーサーとの統合方法の詳細については、SDKのドキュメントを参照してください。

注. Jaeger: Uber社によりGo言語で開発された分散トレーシングのためのオープンソースソフトウェア(https://www.jaegertracing.io/)

質疑応答

THANK YOU

APPENDIX

SDK バージョン

Language	Latest Version	Support Type	
Java	3.x	official	https://docs.couchbase.com/java-sdk/3.0/hello-world/start-using-sdk.html
Node.js	3.x	official	https://docs.couchbase.com/nodejs-sdk/3.0/hello-world/start-using-sdk.html
Python	3.x	official	https://docs.couchbase.com/python-sdk/2.5/start-using-sdk.html
.NET	3.x	official	https://docs.couchbase.com/dotnet-sdk/2.7/start-using-sdk.html
GO	2.x	official	https://docs.couchbase.com/go-sdk/2.0/hello-world/start-using-sdk.html
C	3.x	official	https://docs.couchbase.com/c-sdk/3.0/hello-world/start-using-sdk.html
PHP	3.0	official	https://docs.couchbase.com/php-sdk/3.0/hello-world/start-using-sdk.html
Scala	1.x	official	https://docs.couchbase.com/scala-sdk/1.0/hello-world/start-using-sdk.html
Ruby	3.x	official	https://github.com/couchbase/couchbase-ruby-client/releases
Rust		community	https://github.com/couchbaselabs/couchbase-rs

Support Policy:
https://www.couchbase.com/support-policy