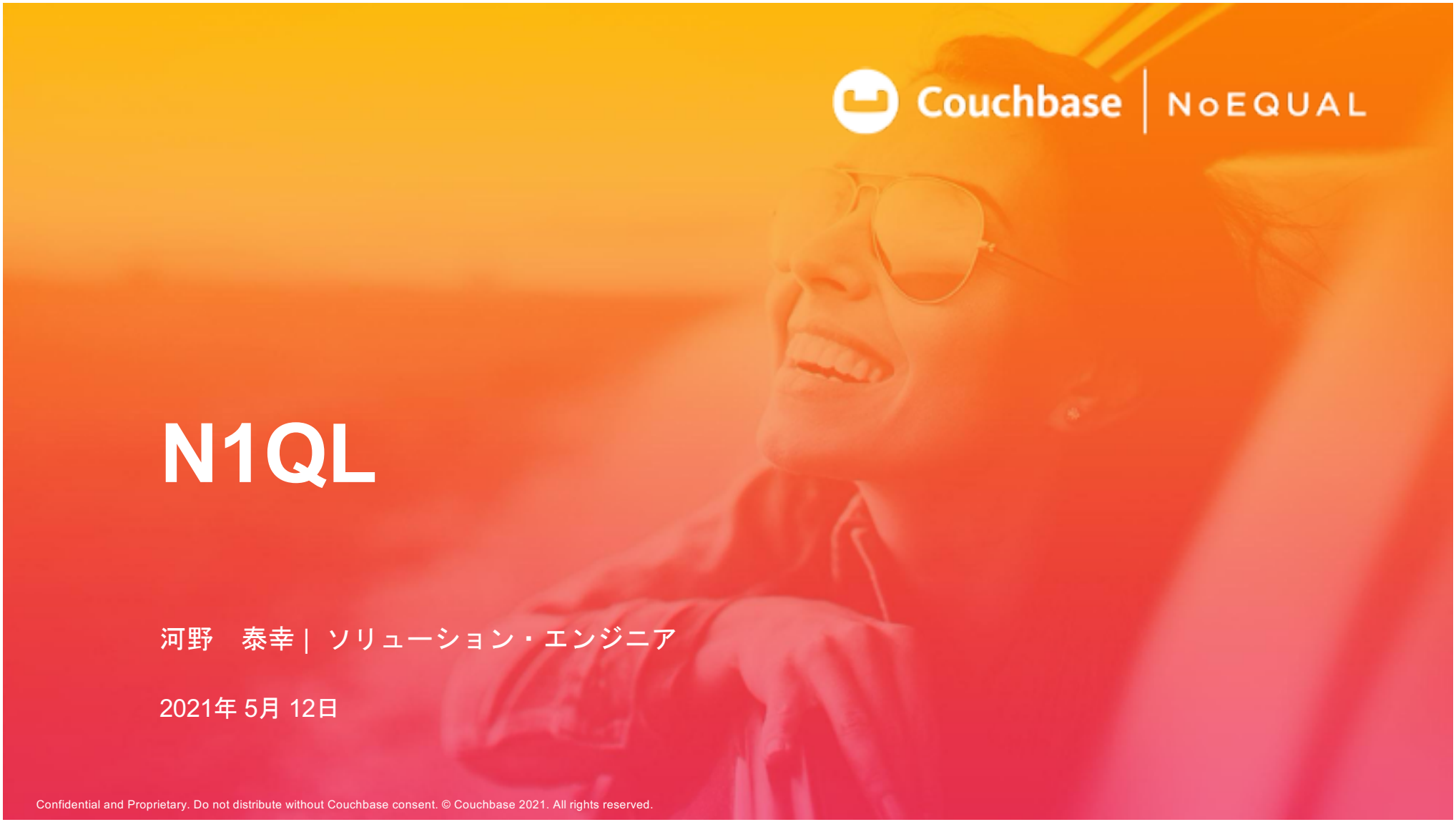


N1QL

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

1. 概要
2. なぜJSONか?
3. N1QL 基礎
4. 配列の操作
5. インデックス
6. N1QL/SQL++ vs MySQL & Mongo クエリ
7. ツール

1 概要

起源

非第 1 正規形クエリ言語
Non **1**st Normalization **Q**uery **L**anguage’

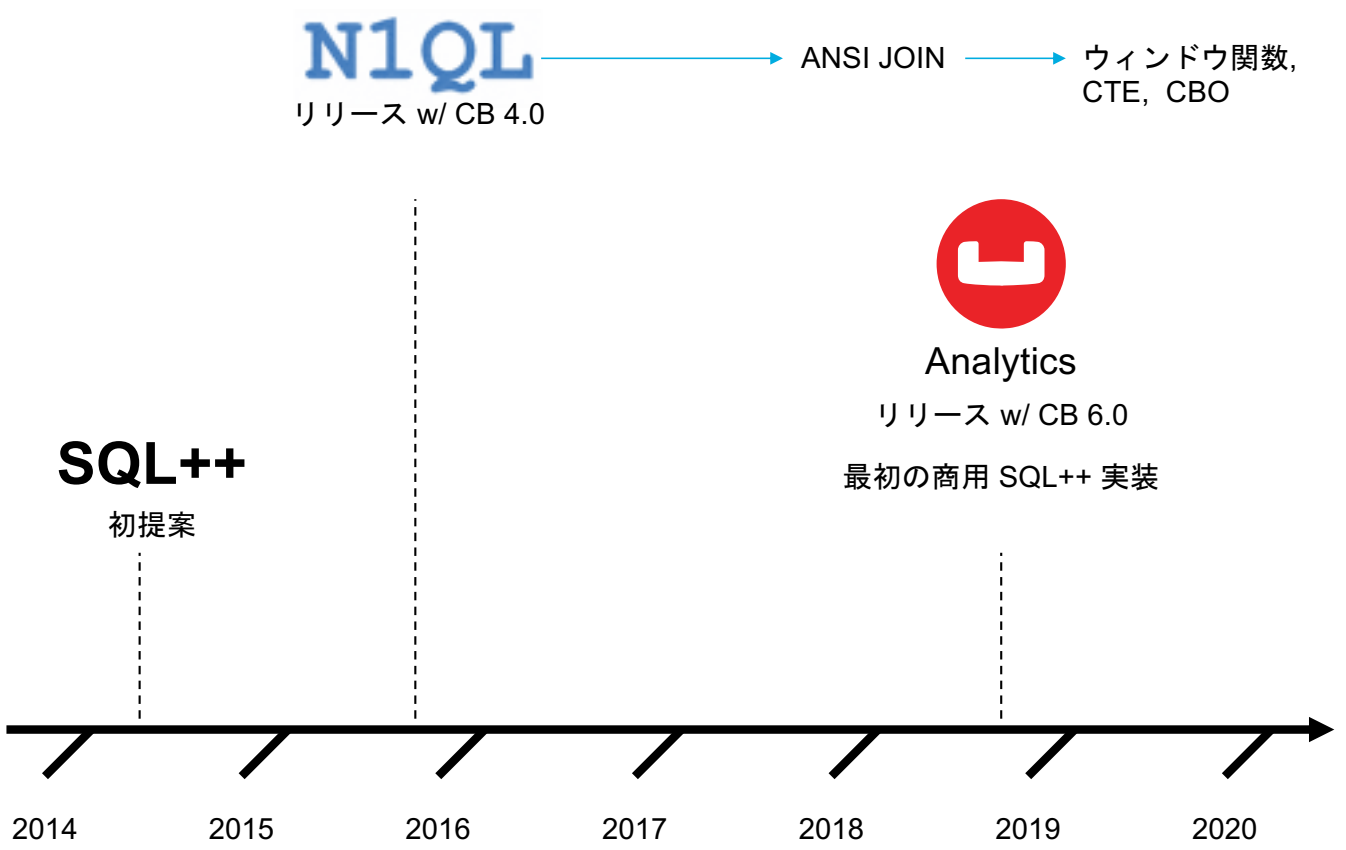
2015 年、Couchbase 4.0 にて初リリース

リレーショナルテーブルではなく、柔軟なJSONドキュメントに対して作用

SQL にできるだけ近くなるように設計

SQLの共同発明者ドン・チェンバレンによる監修

進化



SQL の何が問題か?

NoSQL

SQL に問題はありません
上手く設計され、人気があり、広くサポートされています

問題はリレーショナル構造にあります。

- 厳格, 様々なデータフォーマットをサポートしていません
- 1:N 関係のために再結合が必要
- 「マスター/ディテール」アプリ用に設計

- ボリューム、バラエティ、ベロシティ
- 常に使用可能、コモディティ ハードウェア上で線形にスケーラブル
- 同じデータに対する複数のアクセスパターン

2 | なぜJSONか？

JSON (JavaScript Object Notation)

軽量データ交換フォーマット

人が読み取り可能

解析および生成のために多くの言語に既存のライブラリが存在

Web ページとマイクロサービスに望ましい形式

「子」関係を埋め込む機能

緩和された正規化

リレーショナルDB
強制された
正規化

- データベースによって**強制されるスキーマ**
- テーブル毎に全てのレコードで同じフィールド
- 重複データなし(*正規化*)
- *モノリシックアーキテクチャ*
- **データ登録**に最適化

Couchbase
緩和された
正規化

- スキーマは**データ自体の構造から推測**
- フィールドはドキュメントごとに異なり得る
- データ重複を許容 (*非 正規 化*)
- *スケーラブルなクラスターアーキテクチャ*
- **データ利用**に最適化

埋め込み
または参照?

最初に「**埋め込み**」を検討します。
合理的な理由がある場合に「**参照**」の利用を考えます。

埋め込み

- 所有関係
- 読み取り頻度が、書き込み頻度を大幅に上回る
- データが小さい
- データ毎のわずかな不整合(inconsistency)は問題ない
- 速度に最適化

OR

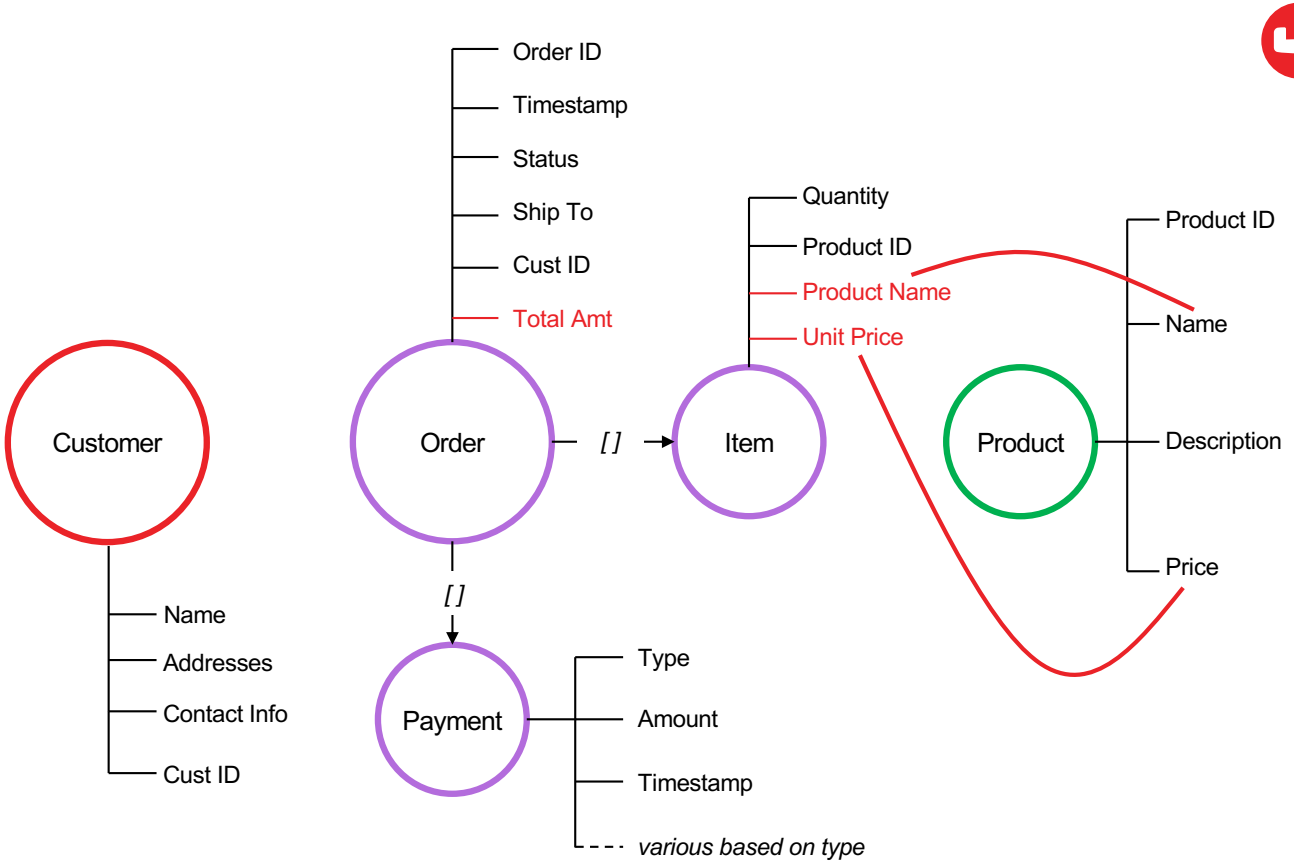
参照

- 非所有関係
- データの整合性(consistency)が重要
- ドキュメントは頻繁に更新される
- キャッシュメモリ利用の最適化
- ドキュメントサイズの削減
- 一意のキーが必要

Couchbase は、以下をサポート:

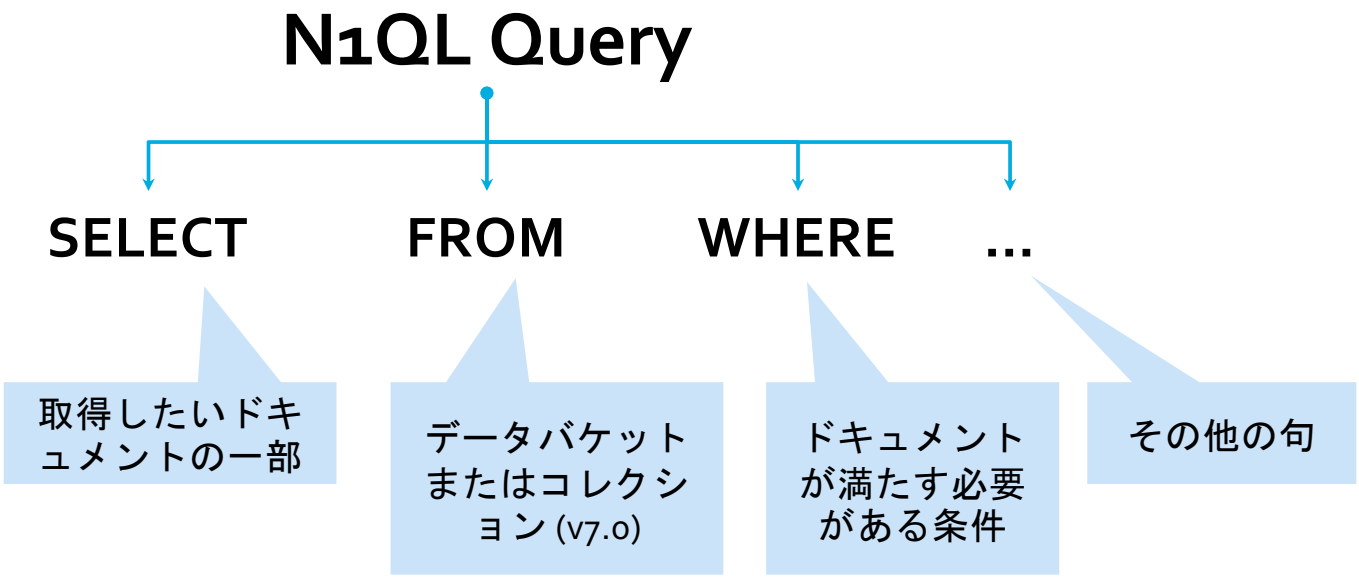
- 単一ドキュメントのアトミックな更新
- サブドキュメントの読み取りと更新
- 複数ドキュメントACID トランザクション

ユースケースに合わせた非正規化

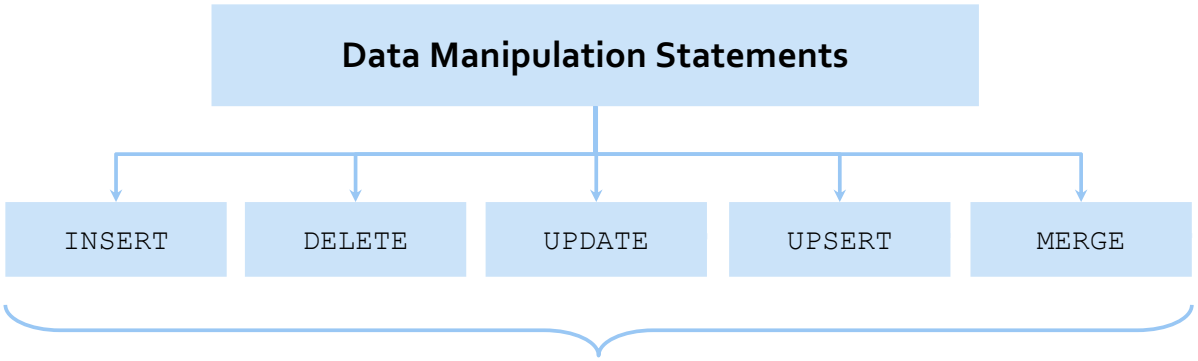


3 N1QL 基礎

見慣れてる?



DML

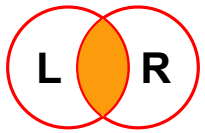


これらのステートメントを使用して、
JSON ドキュメントに保存されているデータを
作成、削除、および変更できます。

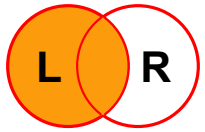
主な機能

インデックス	インデックスの作成と削除には、CREATE INDEX および DROP INDEX ステートメントを使用
アクセス パス	単純なドット表記で埋め込みドキュメントを走査可能
集約	GROUP BY 句、HAVING句のようなグループ化演算子と合わせて、MIN、MAX、COUNTのような集計演算子を提供
結合	複数のドキュメントからなるデータを取得（複数のバケットにまたがる事が可能）
ネストされたクエリ	where 句内に埋め込まれたクエリ内のクエリ

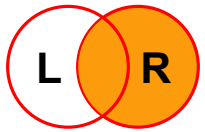
Join オプション



[INNER] JOIN

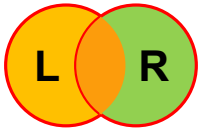


LEFT [OUTER] JOIN

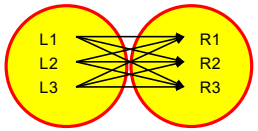


RIGHT [OUTER] JOIN

Join
N1QLにおける制限



FULL [OUTER] JOIN はサポートされていません



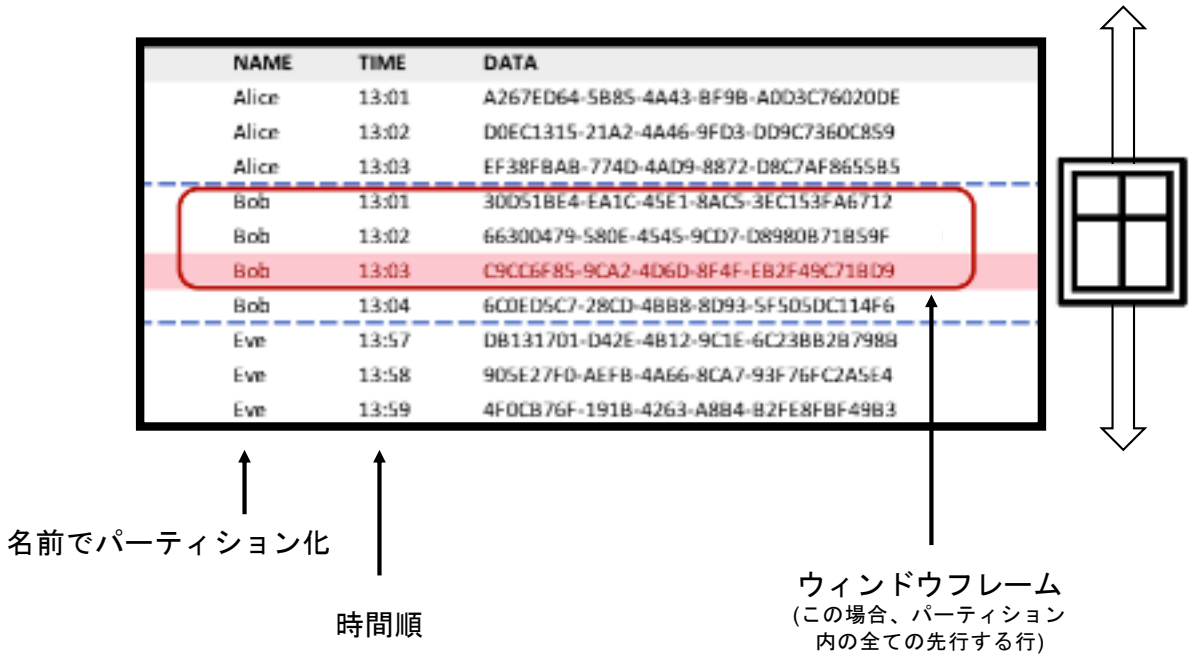
CROSS JOIN はサポートされていません

A RIGHT [OUTER] JOIN は、JOIN連結の最初の (または唯一の) クエリである必要があります

ライトハンド・キースペースには適切なセカンダリインデックスが必要

ウィンドウ関数

ウィンドウ関数は、現在の行に何らかの形で関連するテーブル行のセットに対して計算を実行します(オブジェクトをグループ化せずに集計値または累積値を計算)。



ウィンドウ関数

N1QL クエリでは、ウィンドウ関数は次の部分で利用される:
- SELECT 句
- ORDER BY 句

分析 (SQL++) クエリでは、ウィンドウ関数は以下の箇所でも利用可能:
- WHERE 句
- HAVING 句
- LET 句

- CUME_DIST()
- RANK()
- DENSE_RANK()
- PERCENT_RANK()
- FIRST_VALUE()
- LAST_VALUE()
- NTH_VALUE()
- LEAD()
- LAG()
- ROW_NUMBER()
- RATIO_TO_REPORT()

```
select * from
  (select location.State, location.City, Dollars
  from rentals, location, rates, rates.Rates as agency
  where rentals.Brand = rates.Brand
    and rentals.Date = rates.Date
    and rentals.Rate = agency.Rate
    and rentals.Plan = agency.Plan
    and rentals.LocId = location.LocId
    and rentals.Make = "Honda"
    and rentals.Date >= "2020-02-01" and rentals.Date <= "2020-03-12"
    and rates.Date >= "2020-02-01" and rates.Date <= "2020-03-12"
    group by location.State, location.City
  let Dollars = round(sum(agency.Amount))) a
where rank() over (partition by State order by Dollars desc) = 1
order by State;
```

Common Table Expressions

CTE を使用すると、クエリで複数回再利用されるデータ セットを定義できます。また、複雑なクエリを単純化するために使用することもできます。
最初の SELECT ステートメントの前に宣言する必要があります。

```
WITH current_period_task AS (
  SELECT DATE_TRUNC_STR(a.startDate,'month') AS month,
        COUNT(1) AS current_period_task_count
  FROM crm a
  WHERE a.type='activity' AND a.activityType = 'Task'
    AND   DATE_PART_STR(a.startDate,'year') = 2018
    GROUP BY DATE_TRUNC_STR(a.startDate,'month')
),
last_period_task AS (
   SELECT x.month, x.current_period_task_count,
         LAG(x.current_period_task_count) OVER ( ORDER BY x.month) AS last_period_task_count
   FROM current_period_task x
)

SELECT b.month,
      b.current_period_task_count,
      ROUND(((b.current_period_task_count - b.last_period_task_count ) / b.last_period_task_count),2) AS
MoMChg
FROM last_period_task AS b
```

CTEは…:

- ランタイムの効率を向上
- コードを単純化 (読みやすさ、保守性)

4 配列の操作

配列の操作

Nesting

NEST により、ドキュメント中のサブドキュメントを取得可能

Left hand入力ごとに、一致するRight hand入力が配列に収集され、その結果に埋め込まれます。

Unnesting

UNNEST は NEST の反対

ネストされたサブドキュメントをドキュメントから抽出し、抽出結果を個別のドキュメントにします。UNNEST(ネスト解除)は単一のドキュメントに対して実行されます。

Raw

SELECT RAW は、複数の結果を(ドキュメントとしてでなく)配列に変換します

UNNEST

- ネストされたオブジェクトを最上位ドキュメントとして表示する特殊な種類の JOIN です。
- JSON階層の分解

```
{
  “ordId”: “PT-4897”,
  “status”: “Shipped”
  “items”: [
    {
    “prodId”: “435-289”,
    “qty”: 14
    },
    {
    “prodId”: “893-702”,
    “qty”: 6
    {
    “prodId”: “212-909”,
    “qty”: 2
  ]
}
```

```
SELECT * FROM orders UNNEST orders.items as item
JOIN products AS product
    ON item.prodId = product.prodId
WHERE ….
```

ordId	Status	prodId	qty
PT-4897	Shipped	435-289	14
PT-4897	Shipped	893-702	6
PT-4897	Shipped	212-909	2

取得結果のレコード数は、UNNESTで指定された配列の要素数に拡大される。

NEST

- 外部の子ドキュメントを親の下に埋め込む特別な JOIN
- JSON カプセル化

```
o1:{"order_id":1234,
    "type":"Order",
    "customer_id":"34567",
    "total_price":"65.5",
    "lineitems":["o11","o12","o13"]
}
o11:{"lineitem_id":o11,
    "type":"lineitem",
    "item_id":"789",
    "qty":"3",
    "itemprice":"5.99",
    "base_price":"17.97",
    "tax":"0.75",
    "total_price":"18.22"
}
o12:{"lineitem_id":o12,
    "type":"lineitem",
    "item_id":"234",
    "qty":"5",
    "itemprice":"10.00",
    "base_price":"50.00",
    "tax":"0.75",
    "total_price":"50.75"
}
```

```
SELECT ordr.order_id,
ARRAY {“item_id”: l.item_id, “quantity”:l.qty} FOR l in line END
as items
FROM `retailsample` ordr
NEST `retailsample` line
ON KEYS ordr.lineitems
```

```
[
  {
    "items":[
      {"item_id":"789", "qty":"3"},
      {"item_id":"234", "qty":"5"}
      ];
    "order_id":"1234"
  },
  {
    "items":[
      {"item_id":"899", "qty":"8"},
      {"item_id":"651", "qty":"2"}
      ];
    "order_id":"9812"
  },
]
```

ドキュメントのキーを介した参照関係でデータをモデル化した際に用いる事ができる

RAW

SELECT RAW は、フィールド属性を除去することによって返されるデータの量を減らします。

```
SELECT city
FROM `travel-sample`
WHERE type="airport"
ORDER BY city LIMIT 5;
```

```
[
 {
  "city": "Abbeville"
 },
 {
  "city": "Aberdeen"
 },
 {
  "city": "Aberdeen"
 },
 {
  "city": "Aberdeen"
 },
 {
  "city": "Abilene"
 }
]
```

```
SELECT DISTINCT RAW city
FROM `travel-sample`
WHERE type="airport"
ORDER BY city LIMIT 5;
```

```
[
  "Abbeville",
  "Aberdeen",
  "Abilene",
  "Adak Island",
  "Addison"
]
```

配列関数

ARRAY_AGG(expr)
ARRAY_APPEND(expr, val1, val2, …)
ARRAY_AVG(expr)
ARRAY_BINARY_SEARCH(expr, val, …)
ARRAY_CONCAT(expr1, expr2, …)
ARRAY_CONTAINS(expr, val)
ARRAY_COUNT(expr)
ARRAY_DISTINCT(expr)
ARRAY_EXCEPT(expr1, expr2)
ARRAY_FLATTEN(expr, depth)
ARRAY_IFNULL(expr)
ARRAY_INSERT(expr, pos, val1, val2, …)
ARRAY_INTERSECT(expr1, expr2, ...)
ARRAY_LENGTH(expr)
ARRAY_MAX(expr)
ARRAY_MIN(expr)
ARRAY_MOVE(expr, val1, val2)
ARRAY_POSITION(expr, val)
ARRAY_PREPEND(val1, val2, … , expr)
ARRAY_PUT(expr, val1, val2, …)
ARRAY_RANGE(start_num, end_num step_num])
ARRAY_REMOVE(expr, val1, val2, …)
ARRAY_REPEAT(val, rep_int)
ARRAY_REPLACE(expr, val1, val2 max_int])
ARRAY_REVERSE(expr)
ARRAY_SORT(expr)
ARRAY_STAR(expr)
ARRAY_SUM(expr)
ARRAY_SYMDIFF(expr1, expr2, …)
ARRAY_SYMDIFF1(expr1, expr2, …)
ARRAY_SYMDIFFN(expr1, expr2, …)
ARRAY_UNION(expr1, expr2, …)

https://docs.couchbase.com/server/current/n1ql/n1ql-language-reference/arrayfun.html

5 インデックス

インデックス

インデックスによって、
コレクション内のすべてのオブジェクトを
検索する必要なく、
効率的にユーザー指定の条件を満たす
オブジェクトを検索する事ができる

インデックスの種類

	インデックスの種類	定義
1	**Primary**	フルバケットスキャンをサポートするための、バケット全体のドキュメントキーのインデックス
2	**Named Primary**	クラスター内の複数のプライマリ インデックスを許容するための、割り当てられた名前を持つプライマリ インデックス
3	**Secondary**	フィールド (キーと値のペア) またはドキュメント キーのインデックス
4	**Composite**	複数のフィールドにインデックスを付けます。
5	**Functional**	フィールドに適用される式関数の結果として得られる値のインデックス
6	**Array**	配列フィールドの個々の要素に対するインデックス
7	**Partial**	バケット内のドキュメントのフィルター処理されたサブセットのインデックス
8	**Covering**	ドキュメントのフェッチを必要とせずにクエリに完全に応答するインデックス
9	**Duplicate**	ロード バランシング、スケール、および高可用性をサポートするインデックス作成機能
10	**Adaptive**	クエリの柔軟性を高めるために、すべてのまたは特定のドキュメント フィールドに一般的にインデックスを付ける際の配列化アプローチを提供するインデックス作成機能

さらに…

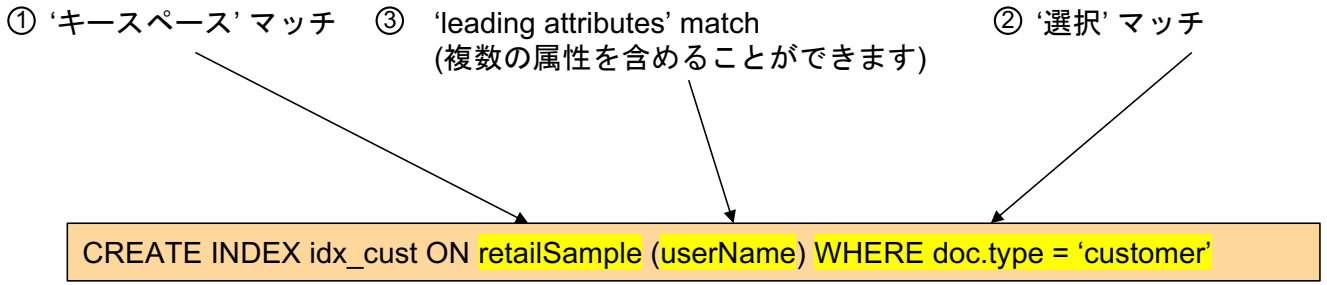
「フルテキスト検索」を使うと、強力な反転インデックスを利用する事が可能。
～ある種のデータモデルとアクセスパターンにおいて、N1QL(GSI)と組み合わせでの活用に効果を発揮

https://docs.couchbase.com/server/current/n1ql/n1ql-language-reference/createindex.html

クエリのための インデックスの 作成

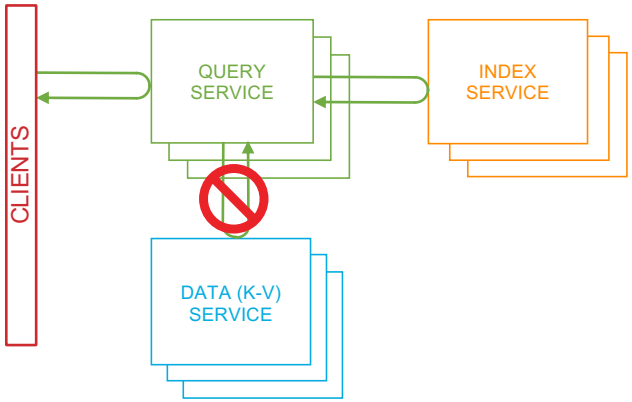
クエリ オプティマイザは、クエリに一致するインデックスを検索します。一致するインデックスが見つからない場合、クエリは失敗します。

```
SELECT c.name.first, c.name.last
FROM retailSample c                    ①
WHERE c.userName = $userName           ③
AND c.doc.type = 'customer'            ②
LIMIT 1;
```



カバリング インデックス

カバリングインデックスとは、クエリの実行を満たすために必要なすべての属性が含まれており、クエリ サービスがデータサービスからドキュメントをフェッチすることを回避できるインデックスです(ただし、インデックス自体が大きくなります)。




```
SELECT c.name.first, c.name.last
FROM retailSample c
WHERE c.userName = $userName
AND c.doc.type = 'customer'
LIMIT 1;
```

```
CREATE INDEX idx_cust ON retailSample (userName, name.last, name.first )
WHERE doc.type = 'customer'
```

カーディナリティはインデックスのパフォーマンスに影響を与える

属性リストの先頭に高いカーディナリティの属性を配置する
（「先行属性(leading attributes)」の一致を妨げない限り）.

```
SELECT c.userName
FROM retailSample c
WHERE c.address.zip = $zipCode
AND c.name.last = $lastName
AND c.doc.type = 'customer'
ORDER BY c.name.last ASC;
```

The order here does NOT matter

'name.last' は'address.zip より選択的です (つまり、高カーディナリティ)
(一般に、「郵便番号より」も多くの種類の「姓」があります)

```
CREATE INDEX idx_cust_lName_zip ON retailSample (name.last, addres.zip)
WHERE doc.type = 'customer'
```

これは一致します

```
SELECT c.userName
FROM retailSample c
WHERE c.name.last = $lastName
AND c.doc.type = 'customer'
ORDER BY c.name.last ASC;
```

これは一致しません

```
SELECT c.userName
FROM retailSample c
WHERE c.address.zip = $zipCode
AND c.doc.type = 'customer'
ORDER BY c.name.last ASC;
```

関数インデックス

インデックス は、計算された値を格納/使用できます（そのインデックスを利用するクエリはその恩恵に預かります）。ただし、 このコンテキストでは、複数ドキュメントに跨がる集計関数 (SUM、AVG、COUNT 等) は使用できません。

```
SELECT c.name.first, c.name.last
FROM retailSample c
WHERE UPPER(c.userName) = $userName
AND c.doctype = 'customer'
LIMIT 1;
```

この値は、大文字として渡されている必要があります。

関数 とパラメータの「**組み合わせ**」が、一致する必要があります。

```
CREATE INDEX idx_cust ON retailSample (UPPER(userName)) WHERE docType = 'customer'
```

配列インデックス

配列インデックスは、配列の要素に関連するクエリを最適化します。配列の要素は、スカラーやネストされた JSON ドキュメントである場合があります。

配列は次の構文を使用してクエリできます。

```
WHERE  EVERY
       o      variable IN expression SATISFIES condition END
       ANY
```

注: 配列をクエリするための高度なオプションが数多くあります。ここでは、混乱を避けるために、基本的な構文のみを示しています。

配列インデックス構文 (シンプル)

配列が単純なスカラー/文字列の場合、単純な式を使用できます。:

```
{
  “doc”: {
    “type”: “product”
  }
  “prodId”: 0267bf1a-7453-4f65-be43-dbe7cde06319,
  “dispName”: “Herman Miller Aeron Ergonomic”,
  “keywords”: [
         ”chair”,
         “ergonomic”
 ]
}
```

ここで[DISTINCT]または[ALL]の
いずれかを選択できます。

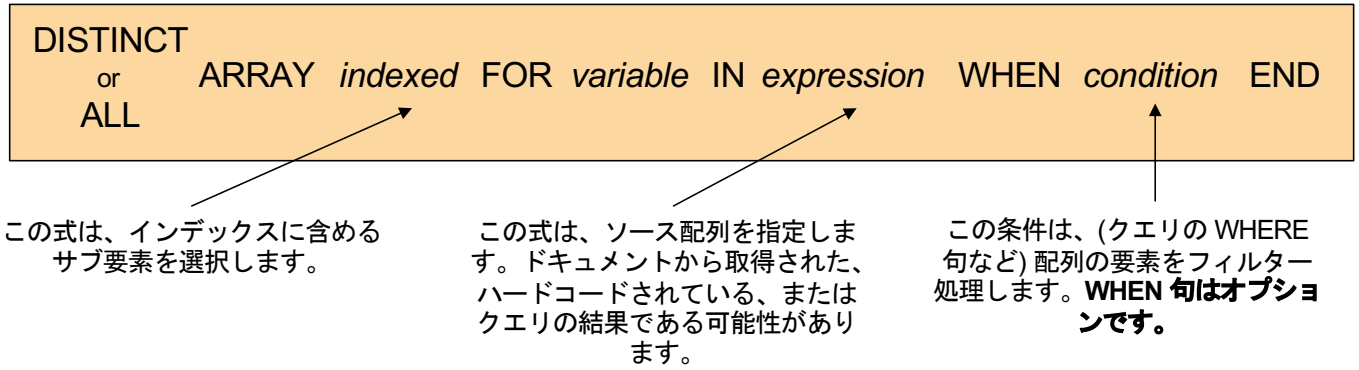
```
CREATE INDEX idx_prod_keyword ON retailSample (DISTINCT keywords)
WHERE doc.type = ‘product’
```

```
SELECT p.prodId AS id, p.dispName AS name
FROM retailSample p
WHERE ANY word IN keywords SATISFIES word = “ergonomic” END
AND p.doc.type = ‘product’;
```

This variable name is local to the ANY…END clause

配列インデックス構文 (完全)

完全な配列インデックス構文を使用すると、サブドキュメントの要素をインデックス化し、含まれる要素をフィルタリングすることができます




```
CREATE INDEX idx_item_product_high_value
ON retailSample (DISTINCT ARRAY item.prodId FOR item IN lineItems WHEN item.price > 100)
WHERE doc.type = 'order'
```

このクエリを使用すると、特定の顧客が購入した高価格製品を取得できます。RAW は結果を配列に格納するので、これをサブクエリとして使用したり、結果を配列として格納したりできます。

```
SELECT RAW prodId AS veryHighValueProds
FROM retailSample o
WHERE ANY item IN lineItems SATISFIES (item.custId = $customerId) AND (item.price > 500) END
AND o.doc.type = 'order';
```

Adaptive Index

ドキュメントのすべて、または指定されたフィールドにインデックスを付けることができる特殊なタイプの配列インデックス
これにより、フィールドのさまざまな組み合わせに対してさまざまな複合インデックスを作成しなくても、効率的なアドホッククエリが可能になります。

```
CREATE INDEX `ai_airport_day_faa`
ON `travel-sample`(DISTINCT PAIRS({airportname, city, faa, type})) WHERE type = "airport";
```

```
SELECT * FROM `travel-sample`
USE INDEX (ai_airport_day_faa)
WHERE airportname LIKE "San Francisco%"
AND type = "airport";
```

```
SELECT * FROM `travel-sample`
USE INDEX (ai_airport_day_faa)
WHERE faa = "SFO"
AND type = "airport";
```

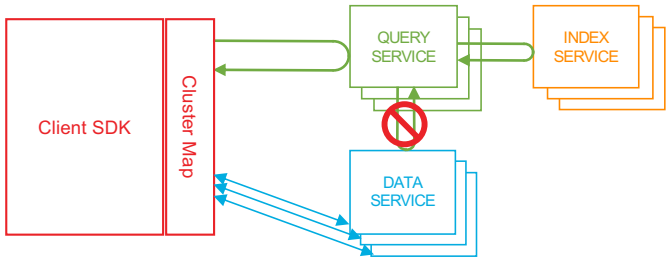
```
CREATE INDEX `ai_self`
ON `travel-sample`(DISTINCT PAIRS(self)) WHERE type = "airport";
```

```
SELECT * FROM `travel-sample`
USE INDEX (ai_self)
WHERE tz = "Europe/Paris" AND (`type` = "airport");
```

https://docs.couchbase.com/server/current/n1ql/n1ql-language-reference/adaptive-indexing.html

複数ドキュメント取得

クエリにより、多数の (かつ大きな) ドキュメントが返される場合は、クエリでドキュメント キーをクライアントに返し、クライアントから K-V API を使用してドキュメントをフェッチすることをお勧めします。これにより、1 つの大きなネットワーク伝送ではなく、多数の小さなネットワーク伝送を用いることになります（すべてのデータ転送を実現するために、クエリプロセスのメモリ領域がボトルネックとなる事がありません。）




```
CREATE INDEX idx_order_pending ON retailSample (meta().id)
WHERE orderStatus = 'Pending'
AND doc.type = 'order';
```

```
SELECT meta().id as docKey
FROM retailSample
WHERE meta().id IS NOT MISSING
AND orderStatus = 'Pending'
AND doc.type = 'order';
```

なぜこれが必要か?
meta().id が欠落している可能性があるか?

「meta().id」が、インデックスの最初の要素であるため。インデックスとの一致のために必要です。

GSI (Global Secondary Index) と FTS (Full Text Search) との比較

GSI または FTS インデックスのいずれもが利用できる場合があります。
それぞれの強みを理解することは、正しいアプローチを決定するのに役立ちます。

機能	GSI	FTS
インデックスに含めるドキュメントに関する選択性(例. パーシャルインデックス)	多くのオプション	少ないオプション
WHERE 句内の複数の属性	最適	属性の順序は考慮されない
カバリングインデックス	最適	可能, インデックスサイズが増大する可能性あり
レンジクエリ	最適	トークン化によって複雑になる可能性がある
ランダムテキスト内の検索	不適切	最適
“ファジー” 検索	制約あり、最適ではない (例: LIKE %foo%)	最適
関連性の順序付け Relevance Ordering	制約あり	フレキシブル
地理空間(Geospatial)クエリ	サポートされていません	最適
トークン化 (言語ステミング含む)	制約あり (Functional Index)	最適, フレキシブル
複数の属性で同じ値を検索する	'OR' サブクエリが必要	最適 (‘all fields’)
‘AND’, ‘OR’, ‘UNION’の複雑な組み合わせ	可能	より簡単なクエリ
配列クエリ	ドキュメント モデルとインデックスに依存- Couchbaseにアドバイスを求めてください	

N1QL- FTS インテグレーション

FTS は、インデックス付きデータを GSI とは異なる方法でインデックス化して格納します。Couchbaseを使用すると、FTS クエリを WHERE 句に埋め込み、サブクエリとして使用できます。

```
SELECT b.name, b.brewery_id
FROM `beer-sample` b
WHERE SEARCH(b.description, 'fruity', {"index":"fts_beer_description"})
AND b.type = 'beer';
```

注: *FTS インデックス範囲が条件によってドキュメントのサブセットに制限されている場合、クエリがインデックスと一致するように、その条件を N1QL WHERE 句に含める必要があります。*

ベスト・プラクティスとして（必須ではありませんが）、上記クエリ例のように、利用する FTS インデックス名を含めることができます。

N1QL- FTS インテグレーション

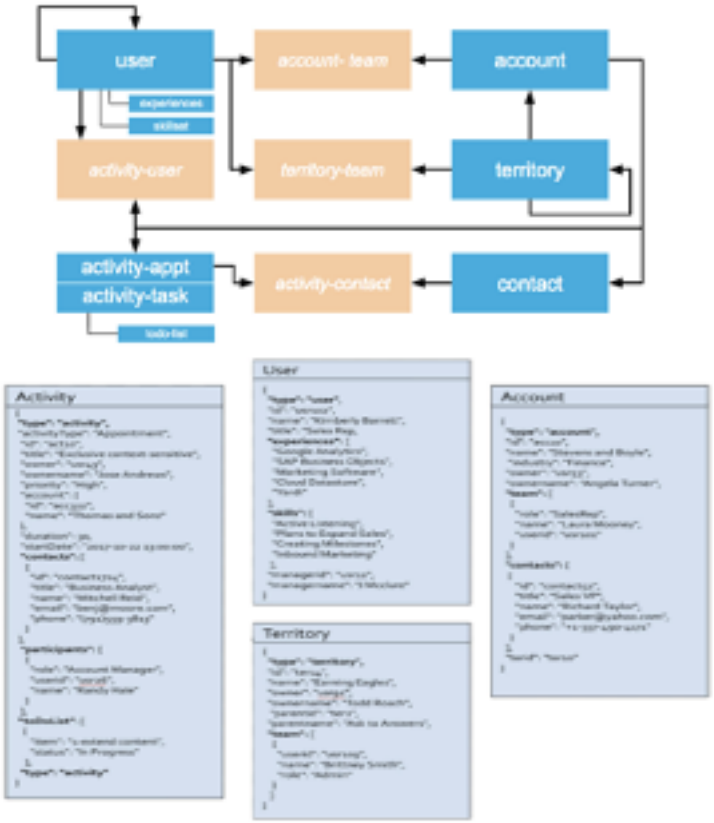
N1QL 検索機能は、より複雑な検索スペックを受け入れることもできます。

```
SELECT b.name, b.brewery_id
FROM `beer-sample` b
WHERE SEARCH(b,
 {“conjucts”:
    [   {”field”:”description”, “match”: ”fruity” },
        {“field”:”category”, “match”: “IPA”}
    ]
 },
 {“index”:”fts_beer_description”})
AND b.type = ‘beer’
AND b.abv > 8.0;
```

6

N1QL/SQL++ vs MySQL & MongoDB クエリ

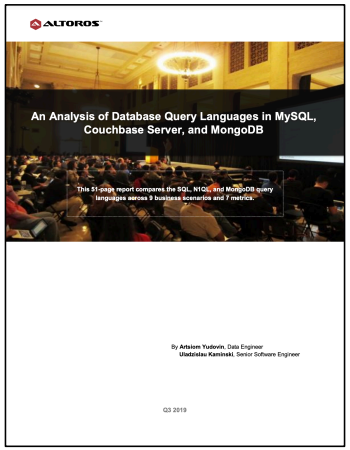
MySQL vs Couchbase vs MongoDB



レポートの目標

1. リレーショナル モデルとドキュメント モデル (CB Team) の両方で表現されるエンタープライズ アプリケーション モデルの使用
2. 一連のビジネス アプリケーションに必要なクエリを作成：Couchbase N1QL (CB チーム)、MongoDB クエリ (Altoros)、 参照としてMySQL用クエリ
3. シンプルさ、読みやすさ、表現力、柔軟性、スキルの可用性に基づいてクエリを比較(Altoros)
4. 主要なメトリックを提供：コード行数、アプリケーションからサーバーへのトリップ数 (Altoros)
5. 調査結果の整理(Altoros)

シナリオ
1. 顧客とのミーティング
2. 地域営業管理
3. 営業活動
4. 営業組織
5. セールス　タスク レポート
6. スキル セット レポート
7. コンタクト検索
8. Google NL API センチメント分析
9. テキスト検索クライテリア



https://info.couchbase.com/rs/302-GJY-034/images/Report_Altoros_Analysis_Database_Query_Languages.pdf

RDBMS からCouchbaseへの簡単な移行

MySQL

```
SELECT ac.industry,
 SUM(CASE WHEN a.activitytype = 'Task'
    THEN 1 ELSE 0 END ) task,
 SUM(CASE WHEN a.activitytype
='Appointment'
    THEN 1 ELSE 0 END ) appts
FROM crm.activity a
  INNER JOIN crm.account ac
     ON (a.accid = ac.id)
WHERE a.startdate BETWEEN '2018-10-01'
           AND   '2018-12-31'
GROUP BY ac.industry
```

Couchbase - N1QL

```
SELECT ac.industry,
  SUM(CASE WHEN a.activityType = 'Task'
    THEN 1 ELSE 0 END) task,
  SUM(CASE WHEN a.activityType =
'Appointment'
    THEN 1 ELSE 0 END ) appts
FROM crm a
  INNER JOIN crm ac ON a.accid = ac.id
          AND ac.type='account'
WHERE a.type='activity'
  AND a.startDate BETWEEN '2018-10-01'
      AND '2018-12-31'
GROUP BY ac.industry
```

MongoDB Query Language

```
db.activity.aggregate(
 { $match: { startDate: { $gt: '2018-01-01',
                          $lt: '2018-12-31' } } },
 {
   $lookup: {
         from: "account",
         localField: "accid",
         foreignField: "id",
         as: "account_docs"
            }
 },
 { $match: { "account_docs": { $ne: [] } } },
 { $unwind: "$account_docs" },
 {
   $project: {
     item: 1,
     task: { $cond:{if:
       { $eq: ["$activityType", "Task"] }, then: 1,
            else: 0 } },
     appt: { $cond: { if:
      { $eq: ["$activityType", "Appointment"] }, then:
1,
            else: 0 } } }
  },
  {
      $group: {
           _id: "$account_docs.industry",
           tasks: { $sum: "$task" },
           appointments: { $sum: "$appt" }
  } }
);
```

Table 3.1.1 Metrics for the Meeting customers scenario

Criteria	MySQL	N1QL	MongoDB query
Simplicity	9	9	9
Readability	10	10	10
Expressiveness	9	9	8
Flexibility	9	9	9
Skills availability	9	7	7
A number of code lines	14	14	38
A number of client/server trips	1	1	1

Table 3.8.1 Metrics for the Calling Google Natural Language API scenario

Criteria	MySQL (unsupported)	N1QL	MongoDB query (unsupported)
Simplicity	-	9	-
Readability	-	10	-
Expressiveness	-	9	-
Flexibility	-	9	-
Skills availability	-	6	-
A number of code lines	-	22	-
A number of client/server trips	-	1	-

Table 3.9.1 Metrics for the Search criteria scenario

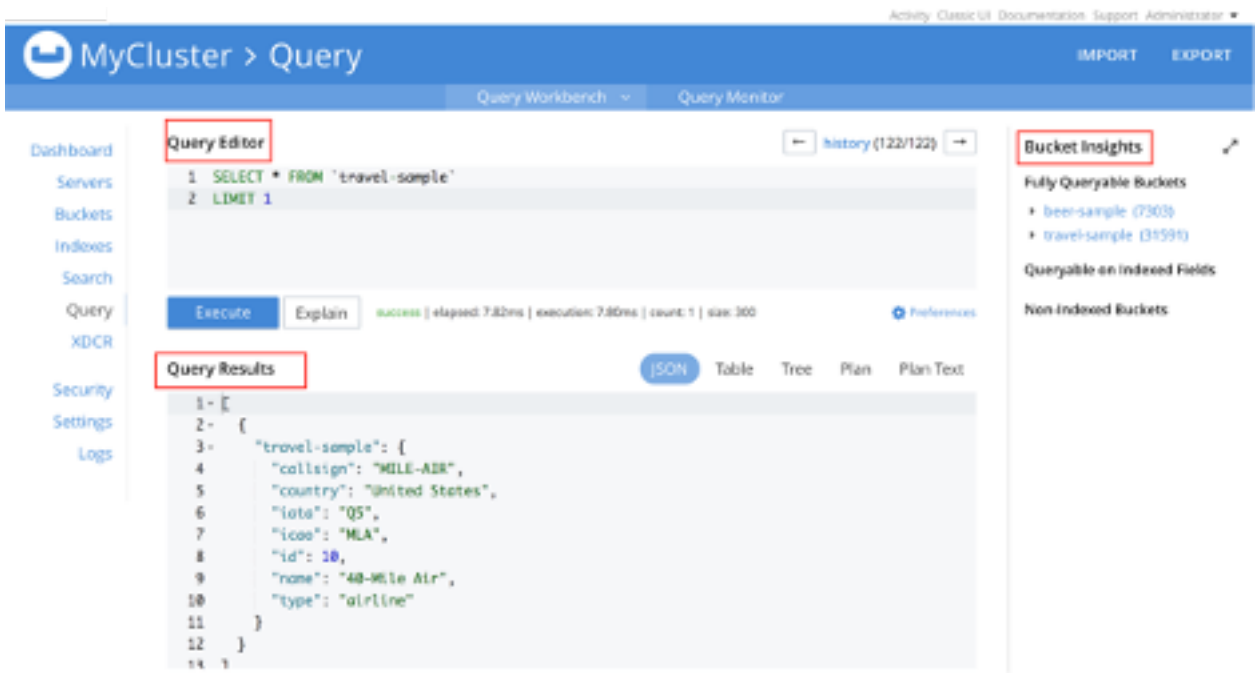
Criteria	MySQL	N1QL	MongoDB query
Simplicity	7	7	7
Readability	8	9	8
Expressiveness	9	9	8
Flexibility	8	10	9
Skills availability	4	6	5
A number of code lines	24	21	26
A number of client/server trips	1	1	1

Listing 3.8.1 An N1QL implementation for the Calling Google Natural Language API scenario

```
SELECT ginfo.name,
 ginfo.review,
 ginfo.sentscore.documentSentiment.magnitude,
 ginfo.sentscore.documentSentiment.score
FROM
 (
  SELECT h.name,
   r.content  review,
   CURL( "https://language.googleapis.com/v1/documents:analyzeSentiment?
           key=YOUR_API_KEY_HERE",
      { "request": "POST",
        "header" :"Content-Type: application/json",
        "data": mydata }
    )  sentscore
  FROM `travel-sample` h
    UNNEST h.reviews r
    LET mydata = '{ "encodingType": "UTF8", "document": { "type":
"PLAIN_TEXT",
                  "content":"' || r.content || '"} }'
  WHERE  h.city = 'Nice'
) ginfo
ORDER BY ginfo.sentscore.documentSentiment.score DESC
LIMIT  10
```

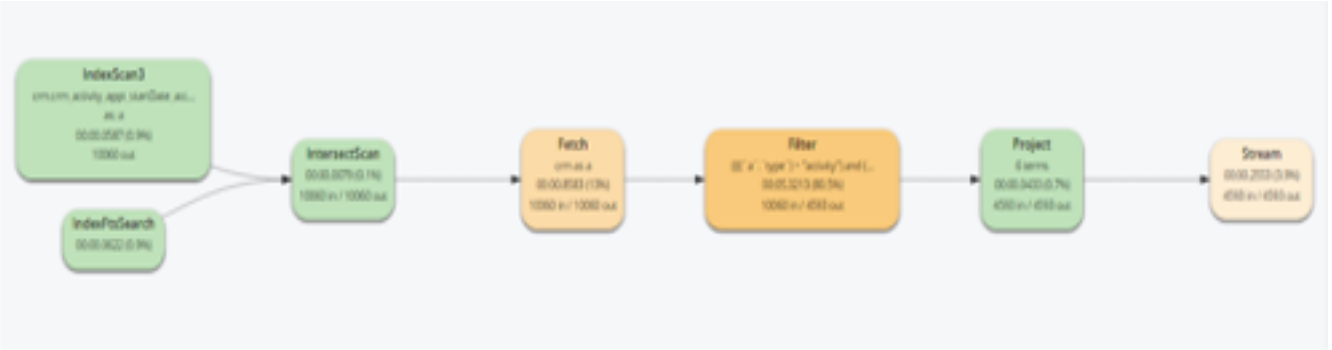
7 ツール

クエリ ワークベンチ



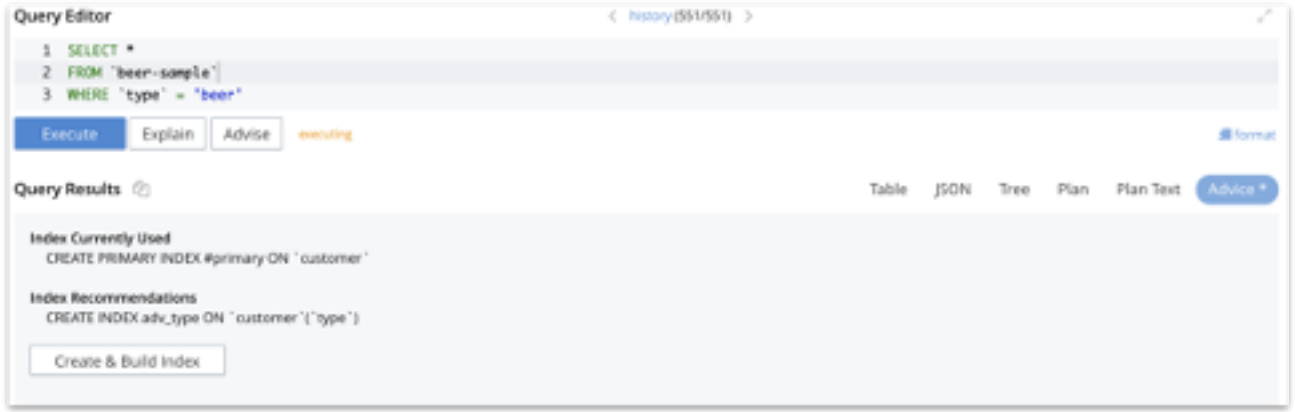
ビジュアル クエリ プラン

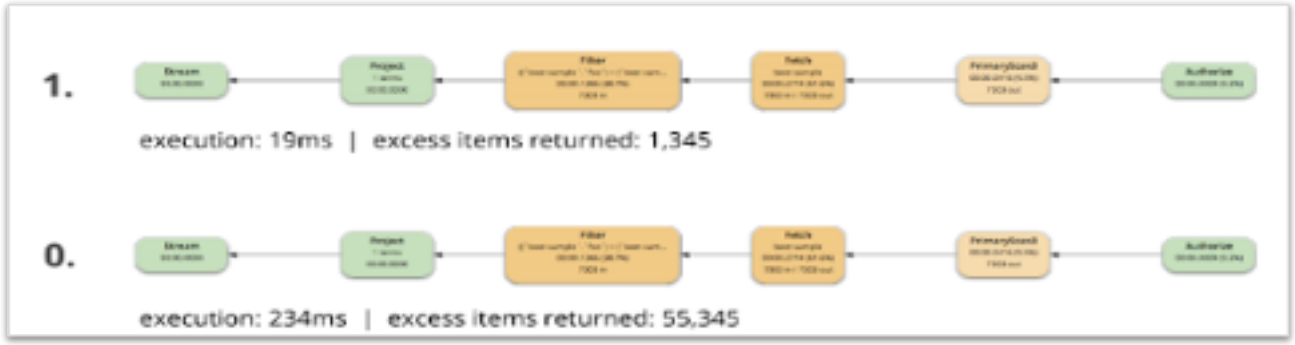
```
SELECT * FROM `beer-sample` b
WHERE SEARCH(b.desc, "fruity")
AND b.abv < 0.5;
```



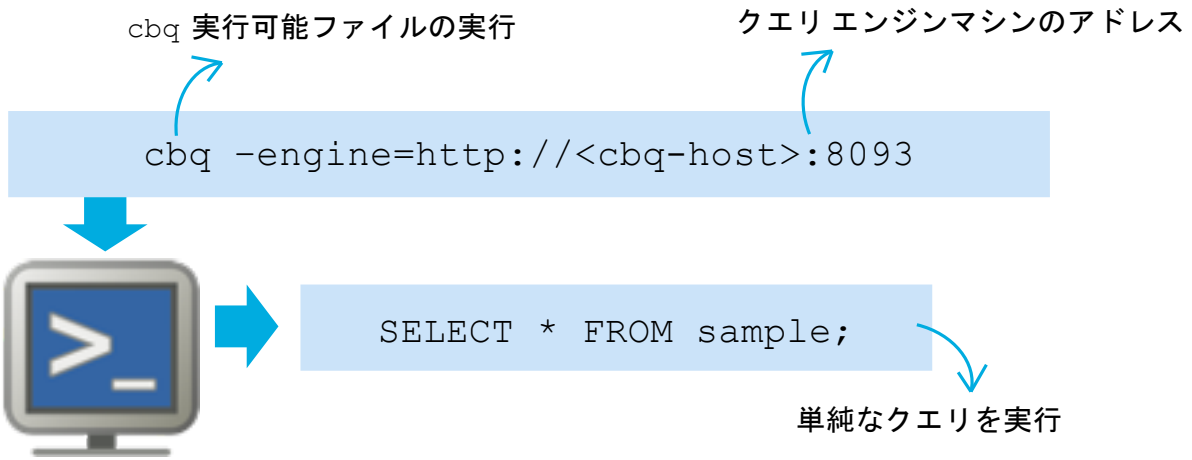
インデックス アドバイザー

インデックス アドバイザーを使用して、クエリに最適なインデックスを作成できます。





コマンドライン



質疑応答

利用可能なその他のリソース

次のセクション: モジュール 5 SDK

Couchbase Download:

https://www.couchbase.com/downloads?utm_source=field_event&utm_medium=workshop&utm_campaign=developerworkshop

For More Information on:

- Couchbase Cloud: https://www.couchbase.com/products/cloud
- Couchbase Academy: https://couchbase.com/academy
  - Certification: https://couchbase.com/academy/certification
- Couchbase Professional Services: https://www.couchbase.com/professional-services

Our Website: http://Couchbase.com