ORACLE

ビジネス / 技術概要

Oracle Databaseによる JSONベース・アプリケーション開発 （およびMongoDB互換性）

リリース19c/21c、オンプレミス/クラウド、
Oracle Autonomous JSON Database、
およびOracle Database API for MongoDB

2022年2月、バージョン1.1

ORACLE

目次

目的 3

柔軟なスキーマを使用したアプリケーション開発 3

NoSQLドキュメント・ストアの制限事項 4

Oracle Databaseのドキュメント・ストアとしての使用 4

OracleにおけるJSONドキュメントの格納と管理 4

Autonomous JSON Database 5

Autonomous Database向けのOracle Database API for MongoDB 5

Simple Oracle Document Access API（SODA） 5

Oracle Databaseに格納されているJSONコンテンツについての分析とレポート作成 7

JSON Dataguide 10

JSONの生成 10

まとめ：Oracle Databaseをドキュメント・ストアとして使用する理由 11

目的

本書では、Oracle Database Release 19cおよび21cの機能と強化された点、ならびに関連のOracleテクノロジーについての概要が説明されています。最新のアプリケーション開発でデータ永続化形式のJSONが頻繁に使用されている理由を把握し、さらに、アプリケーション・データの永続化、問合せ、処理を行うためのドキュメント・ストアを求めている現在の開発者の要求を解決するには、Oracle DatabaseのJSON機能がいかに最適であるかを理解できるよう支援することを目的としています。

柔軟なスキーマを使用したアプリケーション開発

今日のアプリケーション開発は変化しやすい環境下で行われています。ユーザーは、急速に変化するビジネス要件にアプリケーションを適合させ、更新が即座に配布されることを期待しています。つまり、開発者には、アプリケーションを進化させるときに、停止時間やDBAの関与が最小限ですむ柔軟なデータ永続化のメカニズムが必要です。リレーショナル・モデルはこの柔軟性に欠けています。表に静的な'シェイプ'があり、アプリケーションを変更するには、そのシェイプを変更する必要がありますが（新しい列を追加するなど）、それには通常、データベース管理者（DBA）の関与が伴うためです。また、既存データを変更して、新しいスキーマに適合させることが必要な場合もあります。さらには、リレーショナルのアプローチでは、スキーマの事前設計が必要です。アプリケーションのオブジェクト（たとえば'顧客注文'）は、オブジェクトの値を格納する表と列に正規化されます。1つのアプリケーション・オブジェクトが複数の表に対して正規化されることは頻繁にあります。つまり、単純なputまたはget操作には、正しい結合条件を持つすべての参加表が関係するinsertsおよびselectsが必要になりました。開発者はこのマッピングを理解し、SQLを使用してこれを表す必要があります。

このアプローチは、数十年にわたって立証されてはいるものの、最新のアプリケーション開発には、あまりにも厳格かつ形式的で、時間がかりすぎると通常は認識されています。また、大抵はアプリケーションの変更とデータベースの変更を同時に行う必要があるため、停止時間が発生し、運用コストが増加する可能性が高くなります。

ドキュメント・ストア（別名：ドキュメント・データベース）は機能が異なるため、スキーマの事前定義は不要です。代わりに、通常はJSON形式のドキュメントとしてアプリケーション・データをモデル化します。各ドキュメントは自己記述的です（名前付き鍵/値ペアで構成されます）。そのため、値を把握するための外部スキーマは必要ありません。また、ドキュメントはそれぞれ異なる鍵/値ペアを持つことができるため、既存のデータやドキュメントを変更せずに、新しい鍵/値ペアを臨機応変に追加することで、アプリケーションを容易に進化させることができます。データを永続化するためにドキュメントを使用するという方法は、開発者が求める柔軟なストレージ・メカニズムと言えます。

JSONを扱う要求が増しているのは、JSONベースのAPIの普及にも起因しています。RESTサービスはJSONの入力および出力を使用して動作します。サード・パーティ製APIが変更され、表と合致しなくなっている場合、これらのJSON値を表にマッピングすると、アプリケーションの中断が発生する可能性があります。マッピングする代わりに、JSONデータを、JSONデータへの問合せをサポートするデータベースに"そのまま"保存する方が適切です。

NoSQLドキュメント・ストアの制限事項

NoSQL製品はリレーショナル・データベースよりも容易に使用できると認識されているため、開発者はNoSQL製品に引かれる傾向にあります。標準的なNoSQLドキュメント・ストアでは、JSONドキュメントはコレクション内に編成されます。データ・モデルが単純であるため（コレクションとドキュメントのみで構成）、これらのシステムが提供する機能も単純であり、特にレポート作成や分析のユースケースでは、機能が限定されます。要求があれば、開発者はたいてい2つ目の（リレーショナル）データベースを導入し、データを2回保存します。通常、データをリレーショナル形式に変換するETLプロセス（Extract：抽出、Transform：変換、Load：ロード）が必要です。また、NoSQLドキュメント・ストアは、通常は複雑なトランザクションや参照整合性制約に対応していないため、データ整合性が開発者の課題となります。必要な'回避策'を講じると、システムの複雑性が増し、セキュリティが低下し、整合性が損なわれる可能性が生じ、さまざまなデータベースでのポイント・イン・タイム・リカバリといった新たな課題が生じます。この複雑性の増加が原因で、総所有コストが上昇しやすくなり、単純なNoSQL製品という約束を遂行できなくなります。

Oracle DatabaseのJSONドキュメント・ストアとしての使用

Oracle Databaseでは、特定用途のNoSQLドキュメント・ストアと同様にアプリケーションを開発できます。JSONドキュメントの格納、管理、索引付けができ、一般的なNoSQL製品のように、NoSQLに似たドキュメント・ストアAPIが提供されます。もっとも普及しているドキュメント・ストアの1つであるMongoDBと互換性のあるAPIもサポートしています。さらに（しかもNoSQL製品とは異なり）、JSONドキュメントに対する高度なSQL問合せ、レポート作成、分析、機械学習も可能です。そのため、JSONとリレーショナル・データを同じ問合せで結合し、統合することができます。また、JSONの機能はOracle Databaseに統合されているため、Oracle Databaseに搭載されている可用性、セキュリティ、スケーラビリティ、パフォーマンス、管理性のためのエンタープライズ機能はすべて、JSONデータに対してもサポートされます。

「JSONデータベースの、Yahoo! Cloud System ベンチマークに基づく独立したベンチマークでは、オラクルがこの分野の圧倒的なリーダーであり、他のどの競合他社よりもパフォーマンスが優れていることが明らかになりました[...]」

Accenture技術レポート：
『Increase agility and cut development time with JSON and Oracle』（2021年版）
https://accntu.re/3Iezy00

Oracle DatabaseにおけるJSONドキュメントの格納と管理

Oracle Database Release 21cでは、高速問合せとピース単位の更新のために最適化されたバイナリ形式を使用する、新しいSQLデータタイプ'JSON'が追加されました。19cをはじめとするそれ以前のリリースでは、JSONドキュメントは、VARCHAR2、CLOB、またはBLOBの各列を使用して格納できました。列に"IS JSON" SQLチェック制約を設定すると、その列には有効なJSONドキュメント以外は含まれなくなるため、列がJSONドキュメントのコンテナとして使用されていることをデータベース側で把握できます。

オラクルのJSON機能では、柔軟なスキーマを使用した開発とドキュメントベースのストレージを完全にサポートすることが重視されています。したがって、Oracle Database側では、所定の列にJSONドキュメントが含まれることは把握していますが、ドキュメントの格納、索引付け、問合せは、JSONドキュメントの内部構造（鍵/値ペア）を把握せずに実行します。JSONドキュメントの構造は、開発者が必要に応じて自由に変更できます。

Oracle Databaseでは、ディザスタ・リカバリ、レプリケーション、圧縮、暗号化をはじめとする高度な機能すべてでJSONが完全にサポートされます。

また、Oracle GoldenGateやOracle Data IntegratorといったOracle Databaseをサポートする製品でも（さらにはサード・パーティのツールでも）、データベースに格納されたJSONドキュメントがシームレスにサポートされます。

Autonomous JSON Database

Oracle Databaseでは、リリース12.1.0.2以降、JSONがサポートされており、多くのJSON機能がサポート開始後に追加されています。'Oracle Autonomous JSON Database'（Oracle AJD）と呼ばれるマネージド・データベース・クラウド・サービスでは、この技術レポートで概要を説明している機能が、Oracle Autonomous Databaseファミリーの他の製品よりもはるかに安い価格帯で提供されます。Oracle AJDでは、ドキュメント・ストアAPIがサポートされるほか、任意のSQLを実行でき、非JSONデータをリレーショナル表に格納できます。Oracle AJDはJSON開発者を対象にしているため、JSON以外のデータには20 GBまでという制限があります。その制限を引き上げる必要がある場合は、ワンクリックでAutonomous Transaction Processing（ATP）サービスにアップグレードできます。したがって、Oracle AJDは、さまざまなスキルやAPIが必要な別の開発環境ではありません。

Oracle AJDユーザーは、Autonomous Databaseプラットフォームの一部として、Autonomous Databaseの自己稼働、自己保護、自己修復機能の利点を十分に享受できます。データベースの稼働時間は最大化され、自動スケーリング（構成済みのCPU上限の3倍まで）によって、最小限のコストで最大限のパフォーマンスが実現します。

Autonomous JSON Databaseサービスについて詳しくは、以下のリンク先を参照してください。

https://www.oracle.com/jp/autonomous-database/autonomous-json-database//

Autonomous Database向けのOracle Database API for MongoDB

Autonomous JSON Databaseを含むすべてのOracle Autonomous Databaseは、MongoDBと互換性があります。MongoDB用に作成されたツール、ドライバ、アプリケーションは、MongoDB用のネイティブAPIを使用してOracle Autonomous Databaseに接続できます。これによりMongoDBデータベース操作は同等のSQL/JSON操作に透過的に変換され、Oracle Database上で実行されます。MongoDBアプリケーションは、MongoDBサーバーに引き続き接続しているかのように、MongoDB APIを介して通信します。開発者はMongoDBのスキルやツールを使用し続けながら、MongoDBコレクションのJSONデータに対してもSQL文を実行できるようになりました。これにより、JSONデータに対するリアルタイムでのSQL分析や機械学習が可能になります。また、リレーショナル・データからJSONを生成し、その結果をMongoDB互換のコレクションとして公開することもでき、MongoDBアプリケーションで問合せ結果やリレーショナル・データに容易にアクセスできるようになります。

Oracle Database API for MongoDBは、Compass、mongo shell、mongoimport/mongorestoreなどのMongoDBツールもサポートしており、Oracleへの移行を簡素化することができます。

2022年2月現在、Oracle Database API for MongoDBは、初期段階として共有のAutonomous Databaseでのみ利用できます。詳しくは、以下のリンク先を参照してください。
http://docs.oracle.com/en/database/oracle/mongodb-api/mgapi

「*Autonomous JSON*は、*MongoDB*が次の*10*年で実現したいと考えている製品です。完全な*ACID*、完全な並列分析、高速アップデート、オープン・スタンダード・ベース*JSON*、フルインデックス、ブロックチェーンから空間、グラフに至るあらゆるもののサポートなどを実現することによって、オラクルは*MongoDB*開発者が未来に飛躍することを支援しています。」

Wikibon
Mark Staimer氏

Simple Oracle Document Access API（SODA）

'Oracle Database API for MongoDB'は現在、共有のAutonomous Databaseに限定されているため、オラクルはオンプレミスだけでなくクラウド（すべてのOracleクラウド・データベース）でも普遍的に利用できる別のドキュメント・ストアAPIを提供します。Simple Oracle Document Access（SODA）API。このAPIは、柔軟なスキーマを使用したアプリケーション開発をサポートするようゼロから設計されており、MongoDB APIなどの一般的なNoSQLドキュメント・ストアAPIと非常に類似しています。

SODAを使用すると、開発者はSQLを習得しなくても、JSONドキュメントとJSONコレクションを操作できます。コレクションやドキュメントに対するデータベース操作は、SQLからではなく、単純なAPIから直接呼び出すことができます。このAPIは、RESTのほか、一般的なプログラミング言語であるJava、Python、JavaScript（Node.js）、C、PL/SQLで利用できます。SODA for RESTはOracle Rest Data Services（ORDS）の一部であり、REST/HTTP呼出しを行うことができるあらゆる言語から起動できます。Java、Python、Node.js、Cの各ドライバはオープンソースです。

SODAのコンセプト・モデルはMongoDBと非常によく似ています。アプリケーション・オブジェクトはコレクションにJSONドキュメントとして保存されます。ドキュメントは鍵で識別されます。コレクションは名前で特定されます。異種コレクションには非JSONオブジェクト（画像など）を格納できます。複数のコレクションが、クライアント・プログラムが接続するデータベースに常駐します。

単純なCRUD操作（create、read + find、update、delete）では通常、SODAコマンドだけでなく、SQLを使用してドキュメントにアクセスできます。同じJSONデータに、レポート作成、分析、機械学習を容易に実行できます。

RESTおよびJavaの例を使用して、SODA APIについて説明します。SODAドキュメント、およびドライバやチュートリアルへのリンクについては、以下を参照してください。

https://www.oracle.com/database/technologies/appdev/json.html

SODAの例

以下のJavaコードは、コレクション'orders'を作成して、JSONドキュメントを挿入します。その後、SODAがドキュメントに割り当てた一意の鍵（id）を取得します。SODAは、ユーザーが生成した鍵を受け入れることもできます。

```
OracleRDBMSClient client = new OracleRDBMSClient();
OracleDatabase    db = client.getDatabase(conn);
OracleCollection orders = db.admin().createCollection("orders");
OracleDocument    doc = orders.insertAndGet(db.createDocument('{…}'));
String id = doc.getKey();
```

お分かりのように、データベース、コレクション、およびドキュメントが、各機能を公開する関数があるJavaクラスにマッピングされます。

SODA for RESTでは、PUT、POST、GET、DELETEなどのHTTP動詞は、ドキュメントに対するSODA操作にマッピングされます。URLには、データベース・ホスト名と認可資格証明とともに、ドキュメントの鍵またはコレクションの名前が含まれます。SODA for RESTは、Oracle REST Data Serviceであり、認証と認可において、ORDSに依存します。この例では、スペースの理由からこの部分が省略されています。コレクションの作成とドキュメントの挿入という2つの操作には、それぞれREST呼出しが必要です。2つ目の呼出しにより、割り当てられた鍵（id）でHTTPレスポンスが生成されます。

「当社は最新のフリート管理ソリューションに、*Autonomous Database*、*JSON*、*REST*を使用します。*JSON*により、アプリケーションの変更、カスタマイズは簡素化され、モバイル機器でも自然に対応できます。」

Satlog GmbH、CEO、Jürgen Stausberg博士
www.satlog.de/en/home/

```
curl -X PUT http://<authUrlToOrds>/soda/latest/orders

curl -X POST -H "Content-type: application/json"
--upload-file document.json http://<urlToORDS>/soda/latest/orders

{
    "items": [
        {
            "id":"A450557094D04957B36346F630CDDF9A",
            "etag":"C13578001CBBC84022DCF5F7209DBF0E6DFFCC626E3B0400C3",
            "lastModified":"2021-02-09T01:03:48.291462",
            "created":"2021-02-09T01:03:48.291462"
        }
    ],
    "hasMore": false, "count":1
}
```

上記の例は、ドキュメント・ストアが従来型のSQLデータベースとどのように異なるかを示しています。新しいドキュメントは、JSONオブジェクトとしてコレクションに追加されます。新しいドキュメントに含まれるキーには、データベースによって課される制約はありません。また、オブジェクト志向のプログラミング環境に慣れている開発者には、API呼出しの方が簡単です。
注：SODA for RESTとその他の言語ドライバ（Javaなど）の違いは、RESTはステートレスであるために、すべてのREST操作が即座にコミットされる一方で、言語ドライバは、トランザクションをサポートするデータベース接続に依存する点です（複数の操作をアトミック操作にすることができます）。

それでは、SODAを使用してドキュメントを取得しましょう。SODAは当然ながら、鍵によるドキュメントのフェッチをサポートしていますが、データを問い合わせるためのより興味深い方法は、検索条件を満たすすべてのドキュメント（JSONドキュメントとして表されます）を見つけることです。これは例示による問合わせ（QBE）と呼ばれます。以下は、フィールド"region"の値が"north"で、2つ目のフィールド"quantity"の値が10以上のすべてのドキュメントを選択する非常に基本的なQBEです。

```
{"region":"north", "quantity":{"$gte":10}}
```

（skipとlimitを使用して大量の結果をページ分けできます。）

以下のJavaスニペットは、QBE検索を実行し、結果を最初の100ドキュメントに制限し、すべてのドキュメント鍵を表示します。変数'qbe'はQBEの上部で有効です。

```
OracleCollection coll = database.openCollection("orders");
OracleCursor results = coll.find().filter(qbe).limit(100).getCursor();
while (results.hasNext())
{
    OracleDocument doc = results.next();
    System.out.println(doc.getKey());
}
```

RESTパラメータaction=queryは、POSTにQBEリクエストが含まれていることを示します。

```
curl -X POST -H "Content-type: application/json" --data
'{"region":"north", "quantity":{"$gte":10}}'
http://<urlToORDS>/ords/SCOTT/soda/latest/orders?action=query
```

「ネイティブな*JSON*のサポートが重要であったのは、以前は、ピュアプレイの*DBMS*でより効率的な*JSON*管理を行うか、*JSON*データをリレーショナル・データなど他のデータと統合するか、どちらかを選択しなければならなかったからです...現在は、*Oracle Database*が*JSON*の効率性と統合データ管理の両方を備えているので、その選択はもはや必要ありません。」

IDC、
Carl W Olofson氏
bit.ly/nativeJSON_IDC

Oracle Databaseに格納されているJSONコンテンツについての分析とレポート作成

このようにOracle Databaseは、アプリケーション開発におけるNoSQLのドキュメント・ストアのあらゆる利点を備えています。Oracle Databaseを使用する主な利点は、SQLのすべての機能を同じJSONドキュメントにも適用できることです。これが可能なのは、JSONコレクションが通常の表を基盤としているためです。自動的に作成されるこの表には、ドキュメントを格納するJSON列と、一意の鍵（ID）とメタデータ（作成日など）のための追加の列があります。ordersコレクションは、次の表によってサポートされます。

```
SQL> describe " orders"

NAME                 NULL?       TYPE
-------------------- ----------- ----------------------------
ID                   NOT NULL    VARCHAR2(255)

CREATED_ON           NOT NULL    TIMESTAMP(6)
LAST_MODIFIED        NOT NULL    TIMESTAMP(6)
VERSION              NOT NULL    VARCHAR2(255)
JSON_DOCUMENT                    JSON
```

Oracle Databaseでは、JSONとともに使用できる以下の広範なSQL演算子がサポートされます。

IS JSON	式にJSONが含まれるかどうかをテスト
JSON_Value	スカラーSQL値を抽出
JSON_Query	JSON断片を抽出
JSON_Exists	1つまたは複数の条件が合致するかをテスト
JSON_TextContains	JSONフィールドで全文検索
JSON_Table	JSONをリレーショナル・モデルに投影
JSON_Object[Agg]	JSONオブジェクトを生成
JSON_Array[Agg]	JSON配列を生成
JSON_Transform	JSONを変更（たとえば、更新の一環として）
JSON_Mergepatch	2つのJSONオブジェクトをマージ
JSON_Dataguide	JSONをサンプリングしてスキーマを構築

多くの演算子は、パス式に依存してJSONデータ内を操作し、パス条件を使用してフィルタを任意に実行します。演算子とパス式について詳しくは、以下の『JSON開発者ガイド』を参照してください。

https://docs.oracle.com/en/database/oracle/oracle-database/21/adjsn/

「当社では、外部*API*やカスタム・ユーザー拡張機能からの予測不可能なデータを扱う場合は必ず、*JSON*を多用しています。*JSON*での*SQL*分析やブロックチェーンもサポートするドキュメント・ストアとして、*Oracle Database*を採用することにしました。」

Retraced、CTO、
Peter Merkert氏
www.retraced.co

以下の例では、このコレクション/表に発注書が含まれていると想定しています。

```
{
    "PONumber":1600,
    "Reference":"ABULL-20140421",
    "Requestor":"Alexis Bull",
    "User":"ABULL",
    "CostCenter":"A50",
    "Instructions": {
        "name":"Alexis Bull",
        "Address": {
            "street":"200 Sporting Green",
            "city":"South San Francisco",
            "state":"CA",
            "zipCode":99236,
            "country":"United States of America"
        },
        "Phone": [
            {
                "type":"Office",
                "number":"823-555-9969"
            }
        ]
    },
    "Special Instructions":"Counter to Counter",
    "LineItems": […]
}
```

SQLを使ってJSONデータを問い合わせるもっとも簡単な方法は、単純なドット表記法であり、それによってJSON構造をナビゲートして値を選択できます。

```
select      j.PO_DOCUMENT.Reference,
            j.PO_DOCUMENT.Requestor,
            j.PO_DOCUMENT.CostCenter,
            j.PO_DOCUMENT.Instructions.Address.city
from J_PURCHASEORDER j
where j.PO_DOCUMENT.PONumber = 1600;
```

REFERENCE	REQUESTOR	COSTCENTER	SHIPPINGINSTRUCTIONS
ABULL-20140421	Alexis Bull	A50	South San Francisco

JSON_TABLEは、あたかも表であるかのようにJSONにアクセスできるよう、JSONをリレーショナル・モデルに投影するためによく使用されるテーブル・ファンクションです。これには、次のような利点があります。

- リレーショナル・モデルは、ウェアハウス方式の問合せがディメンションであり、ファクトが別のコレクションに格納されている場合は特に、分析問合せに極めて適しています。マテリアライズド・ビューを使用すると、このような結合を'事前計算'することさえも可能です。
- リレーショナル・モデルで動作するツールを、JSONとともに使用できます。たとえば、レポート・ビルダー、ダッシュボード、機械学習を、JSONデータに直接適用できます。
- データ・アナリストはSQL言語やチューニングのスキルを利用できますが、JSONデータを表や適切なカスタム・コードに手動でマッピングする必要はありません。

JSON_TABLEは一連のJSONパス式を使用して、JSONドキュメントのコンテンツをリレーショナル列として仮想表に投影します。JSON_TABLE式は、リレーショナル表を使用する場合と同じ方法で、SQL問合せのFROM句で使用します。次の例では、JSONドキュメントのコレクションに含まれる一連の列を投影しています。JSONパス式ごとにドキュメントから単一のスカラー値が返されるため、ドキュメントごとに仮想表の1行が生成されます。

```
select jt.*
from J_PURCHASEORDER p,
JSON_TABLE(p.PO_DOCUMENT ,'$'columns
      PO_NUMBER NUMBER(10) path '$.PONumber',
      REFERENCE VARCHAR2(30 CHAR) path '$.Reference',
      REQUESTOR VARCHAR2(32 CHAR) path '$.Requestor',
      USERID          VARCHAR2(10 CHAR) path '$.User',
      COSTCENTER VARCHAR2(16 CHAR) path '$.CostCenter',
      TELEPHONE VARCHAR2(16 CHAR) path '$.Instructions.Phone[0].number'
) jt
where PO_NUMBER > 1599 and PO_NUMBER < 1602;

PO_NUMBER  REFERENCE          REQUESTOR     USERID   COSTCENTER   TELEPHONE
1600       ABULL-20140421     Alexis Bull   ABULL    A50          909-555-7307
1601       ABULL-20140421     Alexis Bull   ABULL    A50          909-555-9119

2 rows selected.
```

JSON_TABLEはまた、ネストされた配列を持つJSONドキュメントもサポートします。以下のNESTED PATHは、ネストされた"LineItems"配列に対して反復されます。ネストされた配列全体に適用される間、ネストされた配列の外の値（PO_NUMBER）が繰り返されます。

```
select jt.*
from J_PURCHASEORDER p,
      JSON_TABLE(p.PO_DOCUMENT ,'$'columns
          PO_NUMBER NUMBER(10) path '$.PONumber',
          REFERENCE VARCHAR2(30 CHAR) path '$.Reference',
          NESTED PATH '$.LineItems[*]'columns(
              ITEMNO          NUMBER(16)          path '$.ItemNumber',
              DESCRIPTION     VARCHAR2(32) path '$.Part.Description',
              UPCCODE         VARCHAR2(14) path '$.Part.UPCCode',
              QUANTITY        NUMBER(5,4) path '$.Quantity',
              UNITPRICE       NUMBER(5,2) path '$.Part.UnitPrice')
          )
)jt
where PO_NUMBER > 1599 and PO_NUMBER < 1602;

PO_NUMBER REFERENCE ITEMNO DESCRIPTION UPCCODE QUANTITY UNITPRICE
```

「*Oracle Autonomous Database*は、分析負荷からトランザクション負荷まで、企業の重要なデータベース負荷をすべて自律的に実行できることに加え、*ML*、グラフ、*IoT*、*JSON*などをサポートしており、現在のデータベースの市場において他の製品とは一線を画しています。
それぞれに個別のセキュリティ・プロファイルと管理のための学習曲線がある9つの専門データベースを持つことと、どのようなタイプのデータセットでも自律的に動作する単一のデータベースを持つこととでは、どちらが良いか決まっています。」

Constellation、
Holger Mueller氏
bit.ly/ADB_Constellation

```
1600    ABULL-20140421 1 One Magic Christmas    13131092    9    19.95
1600    ABULL-20140421 2 Lethal Weapon          8539162     5    19.95
1601    ABULL-20140423 1 Star Trek 34           9736600     1    19.95
1601    ABULL-20140423 2 New Blood              4339605     8    19.95
1601    ABULL-20140423 3 The Bat                1313111     3    19.95
1601    ABULL-20140423 4 Standard Deviants      6318650     7    27.95
1601    ABULL-20140423 5 Darkman 2              2519203     7    19.95

7 rows     selected
```

JSON_TABLEを使用すると、任意の複雑なJSON構造をリレーショナル・モデルに投影できます。JSON_TABLE問合せはビューとして公開できます。ビューのすべてのコンシューマは、スカラー値を持つ行と列だけで構成される従来表のように、JSONデータにアクセスできます。この方法では、JSONデータ・モデルをサポートしていないリレーショナル・ツールも使用できます。

「*Autonomous JSON*は、データベースのフットプリントを統合し、アプリケーション開発チームの悩みとなる注意力をそらすような状況を減らし、生産性を向上させます。」

ESG、
Mark Peters氏
bit.ly/AJD_ESG

JSON Dataguide

JSON_TABLEの一般的な用法の1つは、JSONの知識がないユーザーやツールがドキュメントを操作できるようにするリレーショナル・ビューを作成することです。JSON_Dataguideを使用すると、コレクション内のすべてのJSONドキュメントをサンプリングして、フィールド名とデータタイプを特定することによって、ビューの作成を自動化できます。以下の例は、ビュー'order_view'を自動作成する方法を示しています。ビュー定義には、上記と似たJSON_Table式が含まれます。

```
declare
    dg CLOB;        -- この変数は派生したJSONスキーマを格納します
begin
    -- JSON_Dataguideがすべてのドキュメントをサンプリングし、JSONスキーマを構築します
    select JSON_Dataguide(json_document, dbms_json.FORMAT_HIERARCHICAL) into dg from
    orders;

    -- このJSONスキーマを使用して、JSON_TABLEビューを自動作成できます
    dbms_json.create_view('order_view', 'orders', json_document', dg); end;
/
```

JSONの生成

Oracle Databaseは、リレーショナル・データやJSONデータから新しいJSONデータを生成することもできます。たとえば、JSON形式でレポートを生成できます。

次の例は、サンプル表employeesとdepartmentsのデータがどのように結合され、結果が新しいJSONドキュメントとして返されるかを示しています。

```
select JSON_ObjectAgg(d.name VALUE (
    select JSON_ArrayAgg(JSON_Object (e.name))
    from employees e
    where e.department_no = d. department_no)
from departments d;
----------------------------
{"ACCOUNTING":[
        {"name":"CLARK"},
        {"name":"KING"},
        {"name":"MILLER"}
    ],
  "RESEARCH":[
        {"name":"SMITH"},
        {"name":"JONES"},
...
```

SQL/JSONの生成演算子は、異なる従来型SQL問合せに単純に追加されます。これは、繰り返しになりますが、Oracle Databaseでいかに効果的にJSONと表を一緒に使用できるかを表しています。生成したドキュメントをコレクションに挿入して、SODA APIやMongoDB APIがアクセスできるようにすることも可能です。

まとめ：Oracle Databaseをドキュメント・ストアとして使用する理由

この技術レポートでは、コレクション内に格納されたJSONドキュメントを利用して、柔軟なスキーマを使用した開発をサポートするOracle Databaseの機能を紹介しました。現在は多くのNoSQLシステムがこの開発パラダイムをサポートしています。では、なぜNoSQLシステムではなく、Oracle Databaseを使用すべきなのでしょうか。

それは、Oracle Databaseがエンタープライズ・アプリケーション向けに構築されているからです。最新のエンタープライズ・クラスのリレーショナル・データベースでは当然と見なされている以下をはじめとする多くの機能は、標準的なNoSQLドキュメント・ストアには搭載されていません。

- 高度な索引付け、問合せの最適化、パラレル実行
- サイズや期間の制約のない、ACIDに完全に準拠したトランザクション
- データ・マスキングや鍵管理などの高度なセキュリティ機能
- 圧縮やデータ・アーカイブなどのデータ管理機能
- オブジェクトレベルのポイント・イン・タイム・リカバリを備えた堅固なバックアップ機能
- 組込み手続き型言語とサーバー側ファンクション

NoSQLシステムには通常、レポート作成や分析操作の機能はありません。JSONドキュメントの量と価値が増すにつれ、クロスドキュメントのレポート作成や分析を実行する機能の必要性も高まっています。
これまで開発者は、データをNoSQLからエクスポートして複雑なETL（Extract：抽出、Transform：変換、Load：ロード）プロセスを適用し、柔軟なレポート作成をサポートするデータ・ストアで使用できるようにする必要がありました。今では多くのNoSQLシステムで、データ・アクセス用として構造化された表形式が必要であることが認識され始めており、基本的なSQLに似た言語が導入され始めているシステムも中にはあります。しかしOracle Databaseでは、高度なSQL分析機能やスケーラブルなパラレルSQLインフラストラクチャとともに、成熟したISO標準SQLのすべての機能をすぐにでもJSONドキュメント・ストアに対して使用できます。

NoSQLドキュメント・ストアを使用する場合は、データがサイロ化される可能性、すなわちリレーショナル・データを管理するデータベースとJSONドキュメントを管理するデータベースが別々になるという問題にも直面することとなります。JSONドキュメント用として独立したデータ・ストアを使用するということはつまり、JSONとして格納されている情報と組織が管理している他の種類のデータ（通常、リレーショナル・データを含む）の結合が必要になった場合は、基本的なタスクであっても、それを成し遂げるには特殊なアプリケーション・コードの開発とメンテナンスが必要になるということです。

オラクルは、アプリケーション開発者向けに設計されたドキュメント・ストア機能をコンバージド・データベースで提供しており、アプリケーション開発者は、オラクルの成熟したデータベース・プラットフォームの他の利点もすべて活用できます。

「Oracle Autonomous JSON Databaseは、MongoDB Atlasの2.3～3.2倍、AWS DocumentDBの2.0～4.1倍高速です。」

Wikibon、
David Floyer氏
https://bit.ly/AJD_Wikibon

Connect with us

+1.800.ORACLE1までご連絡いただくか、oracle.comをご覧ください。北米以外の地域では、oracle.com/contactで最寄りの営業所をご確認いただけます。

blogs.oracle.com　facebook.com/oracle　twitter.com/oracle