ORACLE®

Oracle Database 12c JSON Document Store

Mark D Drake
Manager, Product Management

Program Agenda

1. **Next Generation Application Development**
2. Oracle 12c as JSON Document Store
3. Simple Oracle Document Access (SODA)
4. SODA for Java
5. SODA for REST
6. Query, Reporting and Analysis on JSON documents
7. Summary

ORACLE

Schema-less Development Concepts

- Store application data as documents rather than tables and columns
  - Document format is typically XML or JSON
- Primary access metaphor for the document is Key/Value
  - Each document is assigned a Unique Key
  - The key is used to store, retrieve or update the entire document
- Database agnostic to the structure of the document
  - Application data model is not baked into the database schema
  - Application developers can change the data model “on-the-fly”
  - Most changes to an application can be deployed without “outages” or DBA support

ORACLE

JSON Document Stores

- Schema-less developers gravitate towards JSON document stores
- JSON Document Store Characteristics
  - Simple, easy to use, document centric API's
  - Indexing and querying of JSON
  - Natural fit for popular RESTFul development techniques
- A number of NoSQL databases provide this functionality
  - MongoDB & CouchDB

ORACLE

The problem with JSON Document Stores

- Difficult to perform effective Analytics and Reporting
  - No standard query language other than limited Query By Example (QBE) capabilities
  - No interface with popular Business Intelligence, Analytics and Reporting tools
  - Limited indexing
  - No support for joins between documents or with other types of data
- Limited or no support for RDBMS Table stakes
  - Concurrency Control,
  - Transactions,
  - Read Consistency
  - High Availability and Disaster Recovery

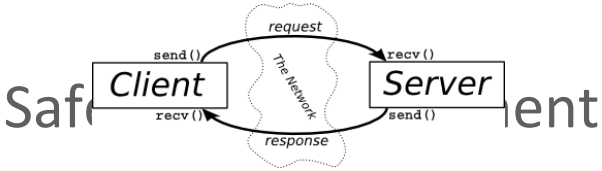
ORACLE

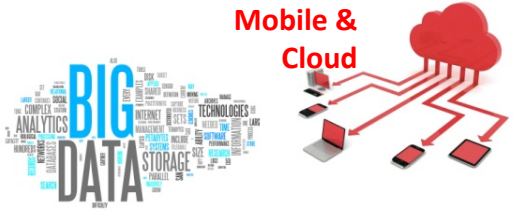
Program Agenda

1. Next Generation Application Development
2. **Oracle 12c as JSON Document Store**
3. Simple Oracle Document Access (SODA)
4. SODA for Java
5. SODA for REST
6. Query, Reporting and Analysis on JSON documents
7. Summary

ORACLE

Application Development Over The Years

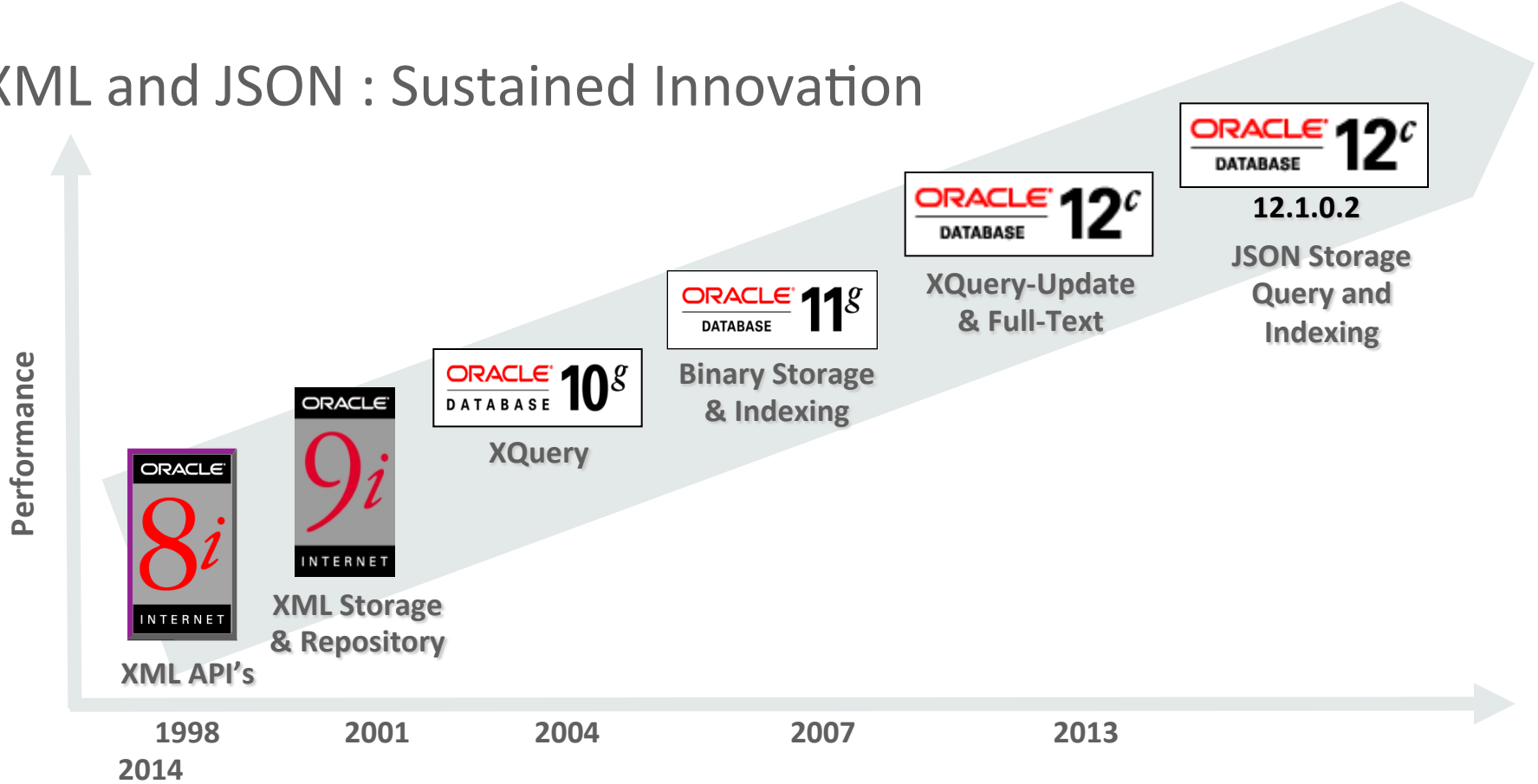




Release	1985 – 1997: 6, 7 and 8	1998 – 2012: 8i, 9i, 10g, 11g	2013 - 2015: 12c
Developer	Stored Procedures & Triggers Referential Integrity Distributed Transactions AQ LOBs Spatial	Java .NET PHP XML APEX	Open Source Drivers (Python, Node.js and R) Pattern Matching OpenSource Drivers JSON REST Data Services NoSQL Database Application Continuity Migration Framework HTML5 – Desktop & Browser Javascript Opensource Cloud
Engine	OLTP throughput (Row Locking, MVRC) Parallel Query Partitioning	Online Operations RAC Data Guard Flashback Self-Managing Database Enterprise Manager Resource Management Automatic Storage Mgmt Encryption Real Application Testing Row Compression Columnar Compression Smart Scans Flash Cache	Multitenant Database In-Memory Column Store

ORACLE

XML and JSON : Sustained Innovation



Enabling ‘No-SQL' development on Oracle Database

- Allow “next generation” application development to leverage Oracle Database
- Allow Oracle Database to store and manage JSON documents
- Provide the “NoSQL” application development experience
  - Simple document centric API’s for storing and accessing JSON
  - QBE Query capability
  - No need to learn SQL
- “NoSQL” means NOT Only SQL.
  - Can still leverage the power of SQL for Reporting and Analytics on JSON documents

ORACLE

Program Agenda

1. Next Generation Application Development
2. Oracle 12c as JSON Document Store
3. **Simple Oracle Document Access (SODA)**
4. SODA for Java
5. SODA for REST
6. Query, Reporting and Analysis on JSON documents
7. Summary

ORACLE

SODA: Goals

- Enable schemaless development on top of an Oracle Database
  - Provide a simple NoSQL-style API for working with documents
- Make it easy to use Oracle as a NoSQL-style document store
  - Allow developers to work with Oracle without learning SQL
  - Allow a developers to work with Oracle without DBA support
- Support all common application development environments
  - Traditional programming languages
  - Scripting languages and frameworks

ORACLE

SODA: Functionality

- Collection Management: Ability to create and drop collections
- Create, Retrieve, Update and Delete (CRUD) operations on documents
- List operations on collections
- Query-by-Example (QBE) for searching collections
- Utility and control functions
  - Create and Drop Indexes
  - Bulk Insert

ORACLE

Other JSON API's

- Implementing SODA style API for NODE.JS, C and C#
- Integrate with popular scripting language and frameworks
  - SODA for REST.
  - Native integrations where necessary

ORACLE

Program Agenda

1. Next Generation Application Development
2. Oracle 12c as JSON Document Store
3. Simple Oracle Document Access (SODA)
4. **SODA for Java**
5. SODA for REST
6. Query, Reporting and Analysis on JSON documents
7. Summary

ORACLE

SODA for Java

- SODA functionality for the Java Developer
- Java Classes for
  - Connection to the database
  - Collection Management
  - CRUD operations
  - Query-by-Example
  - Utility and control functions
- Java implementations of JSON Path processor and JSON Parser
- Much simpler than JDBC for working with collections of JSON documents stored in Oracle Database

ORACLE

Sample SODA for Java code

Creating a Collection, Inserting a Document and getting the ID and Version

```
// Create a Connection
OracleRDBMSClient client = new OracleRDBMSClient();
OracleDatabase database = client.getDatabase(conn);

// Now create a collection
OracleCollection collection = database.getDatabaseAdmin().createCollection("MyCollection");

// Create a document
OracleDocument document = database.createDocumentFromString("{ \"name\" : \"Alexander\" }");

// Next, insert it into the collection
OracleDocument insertedDocument = collection.insertAndGet(document);

// Get the key of the inserted document
String key = insertedDocument.getKey();

// Get the version of the inserted document
String version = insertedDocument.getVersion();
```

ORACLE

SODA for Java Characteristics

- SODA for Java applications use a JDBC connection to talk to the database
- SODA for Java is transactional
  - Supports Hybrid model with JDBC and SODA based operations

ORACLE

Program Agenda

1. Next Generation Application Development
2. Oracle 12c as JSON Document Store
3. Simple Oracle Document Access (SODA)
4. SODA for Java
5. **SODA for REST**
6. Query, Reporting and Analysis on JSON documents
7. Summary

ORACLE

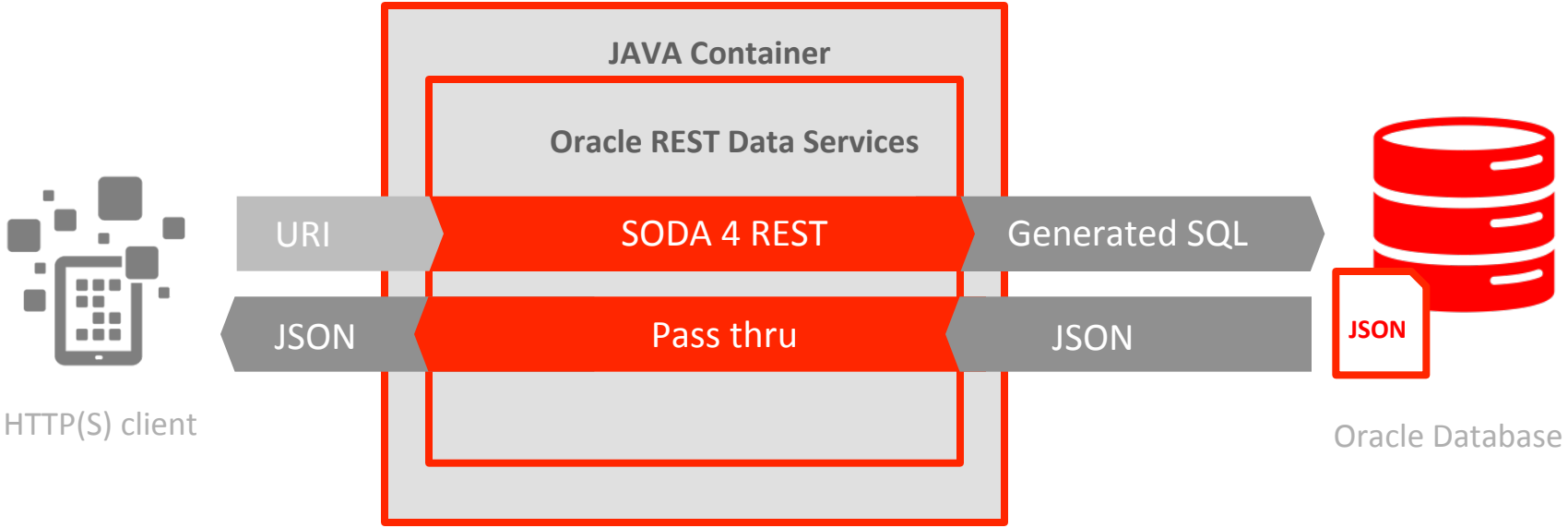
SODA for REST

- RESTFul API for JSON documents stored in Oracle Database 12c
  - Standard REST based model
  - URI patterns mapped to operations on collections managed by the database
- Based on HTTP(s)
  - Can be invoked from almost any programming language
- Implemented as a Java Servlet
  - Run in any container supported by Oracle REST Data Services (ORDS 3.0)
  - Can be installed as a JAVA servlet under the XMLDB HTTP Listener

ORACLE

Oracle REST Data Services

SODA for REST and ORDS 3.0



- Data stored in Oracle Database as JSON documents
- App Developer make standard HTTP(S) calls to SODA for REST API

RESTFul development with JSON

- RESTFul services are a simple well, understood model
- CRUD operations map to HTTP Verbs
  - Create / Update : PUT / POST
  - Retrieve : GET
  - Delete : DELETE
  - QBE, Bulk Update, Utility functions : POST
- JSON document forms the payload of the HTTP Request or Response
- Stateless model, no transaction support

ORACLE

Oracle REST API

GET /rest/schema	List all collections in a schema
GET /rest/schema/collection	Get all objects in collection
GET /rest/schema/collection/id	Get specific object in collection
PUT /rest/schema/collection	Create a collection if necessary
PUT /rest/schema/collection/id	Update object with id
POST /rest/schema/collection	Insert object into collection
POST /rest/schema/coll?action=query	Find objects matching filter in body

ORACLE

JSON Rest Services: Using PUT to Insert a new record

```
PUT /my_database/my_schema/customers HTTP/1.0
Content-Type: application/json
Body:
{
 "firstName": "John",
 “lastName”: "Smith",
 "age": 25,
 "address": {
"streetAddress": "21 2nd Street",
"city": "New York",
       "state": "NY",
"postalCode": "10021“,
"isBusiness" : false  },
       "phoneNumbers": [
       {"type": "home",
        "number": "212 555-1234“ },
       {"type": "fax",
        "number": "646 555-4567“ }  ]
}
```

JSON Rest Services : Invoking Query by Example using POST

```
POST /my_database/my_schema/customers?
command=query HTTP/1.0
Content-Type: application/json
Body:

{
  "company" : "IBM“,
  "$startsWith" : {"department": "S"},
  "$or" : [
    {"$startsWith" : {"name": "Melissa"}},
    {"$gte" : {"salary" : 10000}}
      ]
  }
```

Generated SQL

```
select "JKEY","JVALUE",
        to_char("LAST_MODIFIED",'YYYY-MM-DD"T"HH24:MI:SS.FF'),
        "VERSION"
    from "SCOTT"."EMPLOYEES"
   where JSON_EXISTS(
            "JVALUE",
           '$?( $.company == $B0 && $.department starts with $B1
             && ( $.name starts with $B2 || $.salary >= $B3 ) )'
            PASSING ‘IBM’ AS "B0" ,
                    ‘S’ AS "B1" ,
                    ‘Melissa’ AS "B2" ,
                    10000 AS "B3"
          )
```

Invoking SODA for REST from Javascript

Introducing the XMLHTTPRequest Object

- Javascript routines embedded in an HTML Page can use the XMLHTTPRequest object to interact with SODA for REST
- Defined by the W3C XMLHTTPRequest specification
  - http://www.w3.org/TR/XMLHttpRequest/
- Allows browsers to interact with remote services without refreshing an entire page
- Asynchronous operations prevent browser from blocking while waiting for a response

ORACLE

Fetching a Document (Server assgined ID's)

REQUEST: GET http://server:port/servlet/schema/collectionName/id

```
function getDocument(URL,callback) {
     var XHR = new XMLHttpRequest();
     XHR.open ("GET", URL, true);
     XHR.onreadystatechange = function() {
                                if (XHR.readyState==4) {
                                   callback(XHR,URL);
                                }
                              };
     XHR.send(null);
}
```

ORACLE

Program Agenda

1. Next Generation Application Development
2. Oracle 12c as JSON Document Store
3. Simple Oracle Document Access (SODA)
4. SODA for Java
5. SODA for REST
6. **Query, Reporting and Analysis on JSON documents**
7. Summary

ORACLE

JSON Support in Oracle Database

Powerful SQL Analytics

Oracle Database 12c

JSON

SQL

Data accessed via
RESTful service
or native API's

Data persisted in database
In JSON

Data analyzed via SQL

Oracle JSON SQL Capabilities

- JSON content is accessible from SQL via new operators
  - JSON_VALUE, JSON_TABLE, JSON_EXISTS, IS JSON

```
SELECT ph.phone_number_type, ph.phone_number
FROM customers c,
   JSON_TABLE (
        c.customer_document, '$. phoneNumbers[*]'
        COLUMNS
           phone_number_type   VARCHAR2(10)    PATH '$.type',
           phone_number            VARCHAR2(10)   PATH '$.number'
   ) ph;
```

- JSON operators use JSON Path language to navigate JSON objects
- Syntax developed in conjunction with IBM

JSON : Queries – Simplified Syntax

```
SQL> select j.PO_DOCUMENT
  2       from J_PURCHASEORDER j
  3     where j.PO_DOCUMENT.PONumber = 1600
  4  /

SQL> select j.PO_DOCUMENT.CostCenter, count(*)
  2    from J_PURCHASEORDER j
  3   group by j.PO_DOCUMENT.CostCenter
  4   order by j.PO_DOCUMENT.CostCenter
  5  /

SQL> select j.PO_DOCUMENT.ShippingInstructions.Address
  2    from J_PURCHASEORDER j
  3   where j.PO_DOCUMENT.PONumber = 1600
  4  /
```

ORACLE

Relational projections using JSON_TABLE()

```
SQL> select D.*
  2   from J_PURCHASEORDER p,
  3       JSON_TABLE(
  4         p.PO_DOCUMENT,
  5         '$'
  6         columns(
  7           PO_NUMBER          NUMBER(10)          path  '$.PONumber',
  8           REFERENCE          VARCHAR2(30 CHAR)          path  '$.Reference',
  9           COSTCENTER         VARCHAR2(16)               path  '$.CostCenter'
  10          NESTED PATH '$.LineItems[*]'
  11          columns(
  12            ITEMNO           NUMBER(16)          path '$.ItemNumber',
  13            UPCCODE          VARCHAR2(14 CHAR)   path '$.Part.UPCCode' ))
  14        ) D
  15   where PO_NUMBER = 1600 or PO_NUMBER = 1604
  16 /
```

JSON_TABLE output

1 row output for each member of LineItems array

PO_NUMBER	Reference	Cost Center	ITEMNO	UPCCODE
1600	ABULL-20140421	A30	1	13131092899
1600	ABULL-20140421	A30	2	85391628927
1604	LBISSOT-20141009	A50	1	97366003448
1604	LBISSOT-20141009	A50	2	43396050839
1604	LBISSOT-20141009	A50	3	13131119695
1604	LBISSOT-20141009	A50	4	25192032325

ORACLE

Aggregations over JSON data

```
SQL> select PO_NUMBER po_num,sum(QUANTITY*UNITPRICE) revenue
  2       from J_PURCHASEORDER p,
  3              JSON_TABLE( p.PO_DOCUMENT, '$'
  4                 columns (
  5                   PO_NUMBER number path  '$.PONumber',
  6                   NESTED PATH '$.LineItems[*]'
  7                   columns (
  8                     QUANTITY  number path '$.Quantity',
  9                     UNITPRICE number path '$.Part.UnitPrice')))
 10    group by PO_NUMBER  order by REVENUE desc;

PO_NUM              REVENUE
----------          ----------
7123                1169.35
3589                1153.45
3772                1133.6
2190                1120.35
```

Indexing

- Known Query Patterns : JSON Path expression
  - Functional indexes using JSON_VALUE and, JSON_EXISTS
  - Materialized View using JSON_TABLE()
- Ad-hoc Query Strategy
  - Generalized Inverted Index
    - Based on Oracle's full text index (Oracle Text)
    - Support ad-hoc path, value and keyword query search using JSON Path expressions.

ORACLE

Indexing JSON

```
CREATE INDEX PURCHASE_FTIX_001 ON PURCHASE (doc)
  INDEXTYPE IS CTXSYS.CONTEXT
  PARAMETERS ('SECTION GROUP
               CTXSYS.JSON_SECTION_GROUP
               SYNC (EVERY "SYSTIMESTAMP +  INTERVAL ''1'' MINUTE")
');
```

```
SELECT  * FROM purchase p
  WHERE JSON_TEXTCONTAINS(doc, '$.customerId', '@gmail.com');

-----------------------------------------------------------------------------------------------
| Id  | Operation                    | Name              | Rows  | Bytes | Cost (%CPU)| Time     |
-----------------------------------------------------------------------------------------------
|   0 | SELECT STATEMENT             |                   |  8435 |  4060K|   825   (0)| 00:00:01 |
|   1 |  TABLE ACCESS BY INDEX ROWID| PURCHASE          |  8435 |  4060K|   825   (0)| 00:00:01 |
|*  2 |   DOMAIN INDEX               | PURCHASE_FTIX_001 |       |       |   801   (0)| 00:00:01 |
```

Program Agenda

1. Next Generation Application Development
2. Oracle 12c as JSON Document Store
3. Simple Oracle Document Access (SODA)
4. SODA for Java
5. SODA for REST
6. Query, Reporting and Analysis on JSON documents
7. **Summary**

ORACLE

Summary

Oracle Database 12.1.0.2 supports ‘document store' based application

- Deliver NoSQL style application development experience
- Full support for schema-less development
- Provides value-add capabilities: SQL on JSON, data integration
- Consistent, standard database environment for IT

ORACLE

Oracle JSON Document Store for Application Developers

- JSON documents stored natively in the database using existing data types
  - Choice of VARCHAR2, CLOB and BLOB
  - No need to perform costly object-relational mappings to store and query data
  - All existing Oracle features work with JSON Data
  - Avoid re-inventing the Wheel
- New, simple and intuitive document centric APIs
  - Enables operations on collections and documents without learning SQL
  - Eliminates need for DBA support in order to create and deploy applications
  - SODA for REST provides full support for RESTFul development

ORACLE

Advantages for Application Developers

- Path based queries on JSON documents
- Full power of SQL can be applied to JSON content (Schema-on-Query)
  - Define relational mappings for JSON content using SQL/JSON and JSON path
- Powerful and Flexible indexing options for JSON documents
  - JSON path based Functional Indexes
  - Materialized Views;
  - Full document indexing with inverted JSON index
- SQL and Path based queries automatically optimized to make use of available indexes

ORACLE

Advantages for Application Developers

- Persistent relational access to JSON content via SQL/JSON based views
  - Leverage existing investments in relational Analytical and Reporting tools
- External JSON data sources accessible through external tables
  - JSON in file system (also HDFS) can be accessed via external tables
- All Existing SQL-based API's can access and update JSON content

ORACLE

Development support for Oracle JSON Document Store

Supporting all major development environments and API's

	Native (SQL) API	Document Store API
Java	✓	H1 2015
.NET	✓	Use REST API
Node.js	✓	H1 2015
REST (ORDS)	✓	DB 12.1.0.2
Ruby	✓	Use REST API
Python	✓	Use REST API
PHP	✓	Use REST API
R	✓	Use REST API
Perl	✓	Use REST API

Use new document-store API's to build applications without writing SQL

Existing SQL API's can access JSON documents as well

Oracle JSON Document Store for IT

- Single Infrastructure for data integration and consolidation
  - Majority of organizations already using Oracle to manage mission critical data
  - One technology manages multiple content types: Relational, Spatial, JSON & XML
  - Single language (SQL) can access all of your content
- Core platform:
  - Proven Scalability, Availability, Reliability and Security
  - Backup / recovery
  - Disaster recovery
  - Management tools
  - Multi-tenancy and Consolidation

ORACLE

Advantages of Oracle Database

- Relational model is not going to disappear, Oracle can manage relational and JSON equally well
  - Allows relational data and semi-structured data to coexist
- Oracle delivers a lower TCO than stove-piped, heterogeneous data stores
  - Using Oracle eliminates cost and complexity associated with managing multiple data management platform while providing best-of-breed functionality.
- Oracle Database is easy to install and deploy
  - Multitenant significantly reduces the cost of deploying Oracle Database
  - Self-service applications can spin up a dedicated pluggable database on demand

ORACLE

JSON Support in Oracle Database

Fast Application Development + Powerful SQL Access

Application developers:
Access JSON documents using RESTful

```
PUT /my_database/my_schema/customers HTTP/1.0
Content-Type: application/json
Body:
{
 "firstName": "John",
 “lastName”: "Smith",
 "age": 25,
 "address": {
        "streetAddress": "21 2nd Street",
        "city": "New York",
        "state": "NY",
        "postalCode": "10021“,
        "isBusiness" : false  },
  "phoneNumbers": [
        {"type": "home",
         "number": "212 555-1234“ },
        {"type": "fax",
         "number": "646 555-4567“ }  ]
}
```

Oracle Database 12c

Analytical tools and business users:
Query JSON using SQL

```
select
  c.json_document.firstName,
  c.json_document.lastName,
  c.json_document.address.city
from customers c;

firstName    lastName     address.city
-----------  -----------  --------------
“John”       “Smith”      “New York”
```

Q+A

ORACLE®