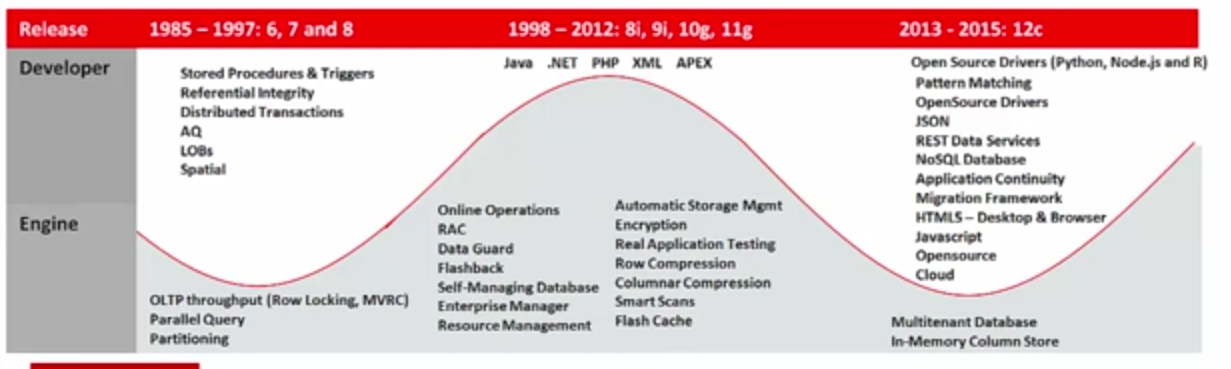
**JSON and Oracle 12c support**

JSON is most useful with simple, structured data. XML is useful for both structured and semi-structured data. JSON is generally data-centric, not document-centric; XML can be either. JSON is not a markup language; it is designed only for data representation. XML is both a document markup language and a data representation language. JSON data and XML data can be used in Oracle Database in similar ways. Unlike relational data, both can be stored, indexed, and queried without any need for a schema that defines the data. Oracle Database supports JSON natively with relational database features, including transactions, indexing, declarative querying, and views.

Unlike XML data, which is stored using SQL data type XMLType, JSON data is stored in Oracle Database using SQL data types VARCHAR2, CLOB, and BLOB. Oracle recommends that you always use an is\_json check constraint to ensure that column values are valid JSON instances

**Image below taken from:** [**https://www.youtube.com/watch?v=4YbH29A\_4gI**](https://www.youtube.com/watch?v=4YbH29A_4gI)





**Based on Oracle Magazine (free – just subscribe online) <http://www.oracle.com/technetwork/issue-archive/2015/15-jan/o15dba-2372211.html>**

Case study:

Acme has been accepting transactions from business partners such as convenience stores and third-party billing companies—and even partners outside the country. To simplify and expedite the processing of transactions from these external systems, Acme has allowed the transactions to come in a **semistructured manner**, in the form of **JavaScript Object Notation (JSON)**—a leading document interchange format (simpler than XML). JSON enables any set of data to be transmitted immediately, without a predetermined format expected by a relational database, and this makes it attractive for integrating outside transactions quickly. Acme’s partners can send any pertinent data they want without first waiting for a mutually agreeable format. However, even though Acme allows data to come in as JSON, the data is stored in a structured manner in the database, in a relational format.

JSON is a standard for free-format text in which any kind of data can be included, along with a descriptor. The descriptor for the data is called a key, and the actual data is called a value. The collection of related data is put into a single JSON document file. Any type of data can be represented as key/value pairs.

**Oracle uses CLOB to store JSON and a constraint IS JSON.**

--

-- Table ACCOUNTS

--

create table accounts (

accno number,

accname varchar2(20),

open\_dt date,

acctype varchar2(1)

)

insert into accounts values (101,'John Smith',sysdate-500,'S')

insert into accounts values (102,'Jane Smith',sysdate-400,'S')

insert into accounts values (103,'John Doe',sysdate-300,'C')

insert into accounts values (104,'John Doe',sysdate-450,'S')

insert into accounts values (105,'Jane Doe',sysdate-200,'C')

--

-- Table TRANSACTIONS

--

create table transactions

(

trans\_id number,

accno number,

trans\_type varchar2(1),

trans\_dt date,

trans\_amt number,

trans\_mode varchar2(20)

)

insert into transactions values (1, 101, 'D', sysdate - 100, 1000, 'Check')

insert into transactions values (2, 101, 'D', sysdate - 150, 2000, 'ATM')

insert into transactions values (3, 101, 'W', sysdate - 90, 1500, 'Transfer')

insert into transactions values (4, 102, 'D', sysdate - 100, 1200, 'Check')

insert into transactions values (5, 102, 'W', sysdate - 200, 1100, 'Check')

insert into transactions values (6, 103, 'D', sysdate - 150, 2000, 'ATM')

insert into transactions values (7, 103, 'D', sysdate - 120, 2500, 'Check')

insert into transactions values (8, 103, 'W', sysdate - 80, 1000, 'Check')

--

-- Table for TRANSACTIONS in JSON Column

--

Note: SYS\_GUID()

Generates and returns a globally unique identifier (RAW value) up to 16 bytes. On most platforms, the generated identifier consists of a host identifier, a process or thread identifier of the process or thread invoking the function, and a nonrepeating value (sequence of bytes) for that process or thread.

create table json\_trans

(

id **raw(16)** not null primary key,

load\_dt timestamp(6) with time zone,

trans\_msg **clob**,

constraint check\_json check (trans\_msg is json) )

insert into json\_trans

values

(

sys\_guid(),

systimestamp,

'{

"accountNumber":101,

"accountName":"John Smith",

"accountType":"Savings",

"openingDate":"2015-04-24T16:33:13",

"transactions":[

{

"transID":1,

"transDate":"2016-05-28T16:50:42",

"transType":"Deposit",

"transMode":"Check",

"transAmount":1000

},

{

"transID":2,

"transDate":"2016-04-08T16:55:56",

"transType":"Deposit",

"transMode":"ATM",

"transAmount":2000

},

{

"transID":3,

"transDate":"2016-06-07T16:55:56",

"transType":"Withdrawal",

"transMode":"Transfer",

"transAmount":1500

}

]

}

'

)

insert into json\_trans

values

(

sys\_guid(),

systimestamp,

'{

"accountNumber":102,

"accountName":"Jane Smith",

"accountType":"Savings",

"openingDate":"2015-08-02T16:38:06",

"transactions":[

{

"transID":4,

"transDate":"2016-05-28T16:55:56",

"transType":"Deposit",

"transMode":"Check",

"transAmount":1200

},

{

"transID":5,

"transDate":"2016-02-18T16:55:56",

"transType":"Withdrawal",

"transMode":"Check",

"transAmount":1100

}

]

}'

)

/

insert into json\_trans

values

(

sys\_guid(),

systimestamp,

'{

"accountNumber":103,

"accountName":"John Doe",

"accountType":"Checking",

"openingDate":"2015-11-10T16:38:06",

"transactions":[

{

"transID":6,

"transDate":"2016-04-08T16:55:56",

"transType":"Deposit",

"transMode":"ATM",

"transAmount":2000

},

{

"transID":7,

"transDate":"2016-05-08T16:55:56",

"transType":"Deposit",

"transMode":"Check",

"transAmount":2500

},

{

"transID":8,

"transDate":"2016-06-17T16:55:56",

"transType":"Withdrawal",

"transMode":"Check",

"transAmount":1000

}

]

}'

)

Retrieving from JSON

SELECT

json\_value(trans\_msg,'$.accountNumber') account\_no,

json\_value(trans\_msg,'$.accountName') cname,

json\_value(trans\_msg,'$.accountType') acc\_type,

json\_value(trans\_msg,'$.transactions[0,1].transID') tran\_id

from json\_trans;