





About me

Experience since 1997

Discovered PL/SQL in 1999

Oracle ACE since 2011

Oracle ACE Director since 2019

Modern Oracle Database Programming 2023

bar-solutions.com
blog

<http://blog.bar-solutions.com>

All things
ORACLE
OTECH
MAGAZINE

<http://allthingsoracle.com>

<http://www.otechmag.com>

[www.red-gate.com/
simple-talk/author/
patrick-barel/](http://www.red-gate.com/simple-talk/author/patrick-barel/)

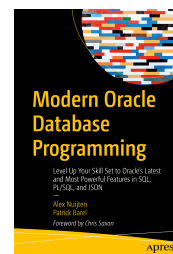
[bar-solutions.com/
otechmagazine.php](http://bar-solutions.com/otechmagazine.php)

SQL

PL/SQL

ORACLE
ACE

ORACLE
ACE Director



Patrick Barel



PBarel@Qualogy.com

Patrick.Barel@GMail.com

patrick@bar-solutions.com



@patch72



Modern Oracle Database Programming

Level Up Your Skill Set to Oracle's Latest
and Most Powerful Features in SQL,
PL/SQL, and JSON

Alex Nuijten
Patrick Barel

Foreword by Chris Saxon

apress®



The Oracle ACE Program

400+ technical experts helping peers globally



- The Oracle ACE Program recognizes and rewards community members for their technical and community contributions to the Oracle community
- 3 membership levels: Director, Pro, and Associate
- Nominate ~~yourself~~ or a colleague at ace.oracle.com/nominate
- Learn more at ace.oracle.com

✉ aceprogram_ww@oracle.com

Facebook.com/OracleACEs

X @oracleace

in Oracle ACE Program Group



ACE Member Benefits



Key Benefits

Cool **swag***, Digital awards for social media, Oracle CloudWorld pass*, & more



Exclusive Content

Exclusive **monthly virtual meetings** with product development teams + engaging guest speakers



Direct Access to Product Management

Multiple **direct communication channels** to product management and fellow ACEs



Networking

In-person & virtual networking opportunities for ACEs to **connect with product development** and each other.



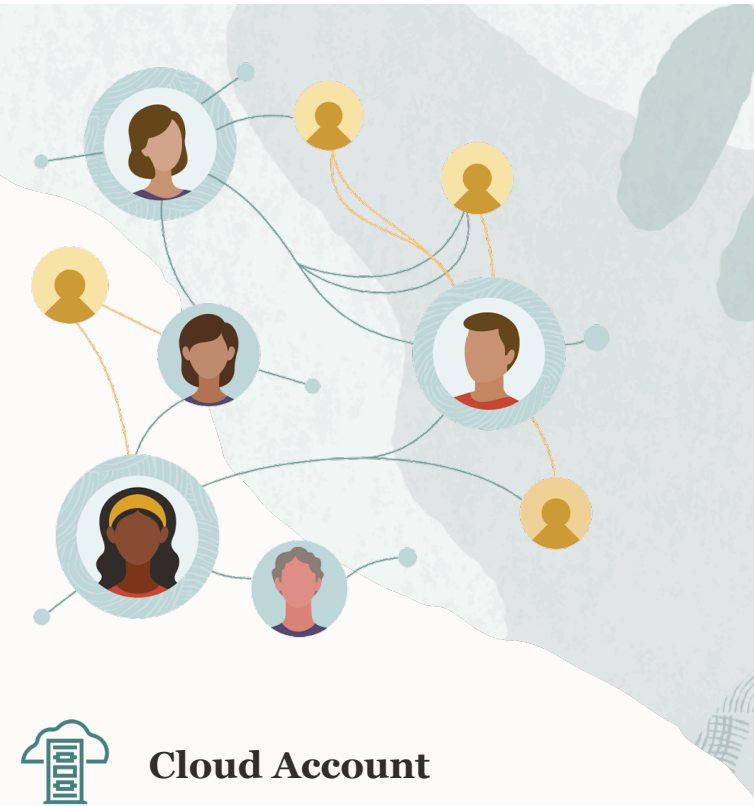
Cloud Account

\$5k USD Cloud account*



Travel Support

ACE Directors are eligible for travel support to give presentations or lead workshops at conferences globally



Oracle Cloud Infrastructure

New Free Tier

oracle.com/cloud/free



Always Free

Services you can use for unlimited time

+

30-Day Free Trial

Free credits you can use for more services





Mentor and Speaker Hub

Our goal is to *connect* speakers with mentors to assist in *preparing* technical sessions and *improving* presentation skills

Interested? Read more and get in touch

<https://mashprogram.wordpress.com>

SYMPOSIUM^{L2}

Created by the community, to support the community

Sharing of reliable knowledge

Supporting the various user groups and individuals



@sym_42



<https://sym42.org/>

JSON Regular Data Variables

Patrick Barel, Qualogy

October 16, 2024



QUALOGY



Both models have their own benefits

JSON

Easy mapping

Schema-less

Hierarchical

Everything combined



Relational

Space efficiency

Flexibility

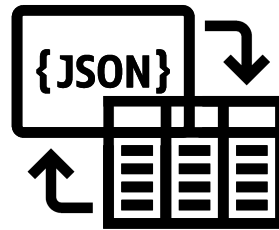
Queryability

Consistency



JSON Relational Duality View

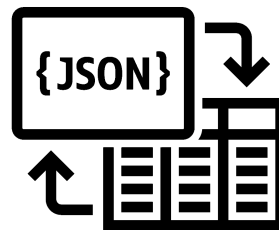
Easy mapping
Schema-less
Hierarchical
Everything combined



Space efficiency
Flexibility
Queryability
Consistency

JSON Relational Duality View

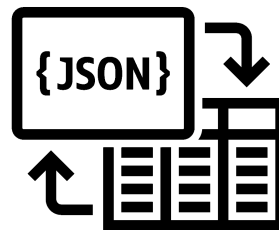
Easy mapping
Schema-less
Hierarchical
Everything combined



Space efficiency
Flexibility
Queryability
Consistency

JSON Relational Duality View

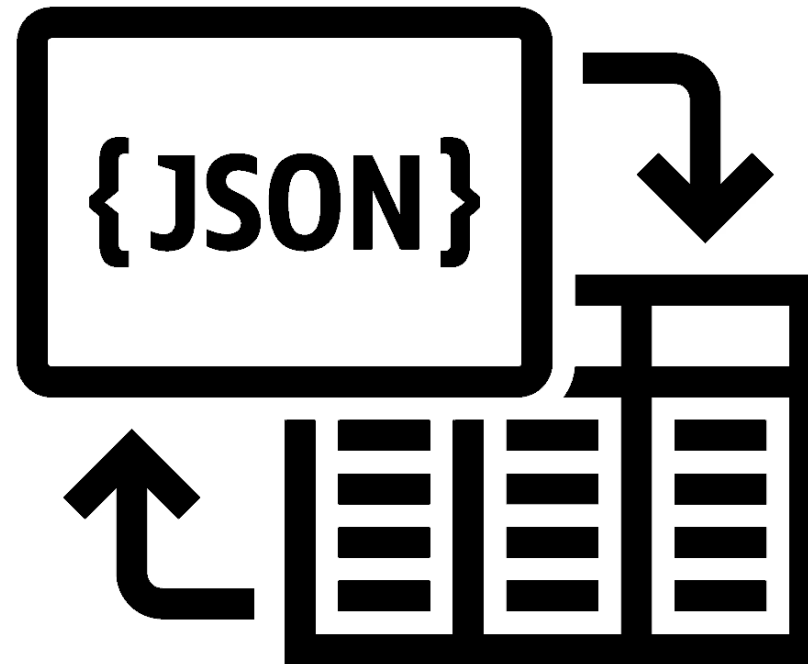
Easy mapping
Schema-less
Hierarchical
Everything combined

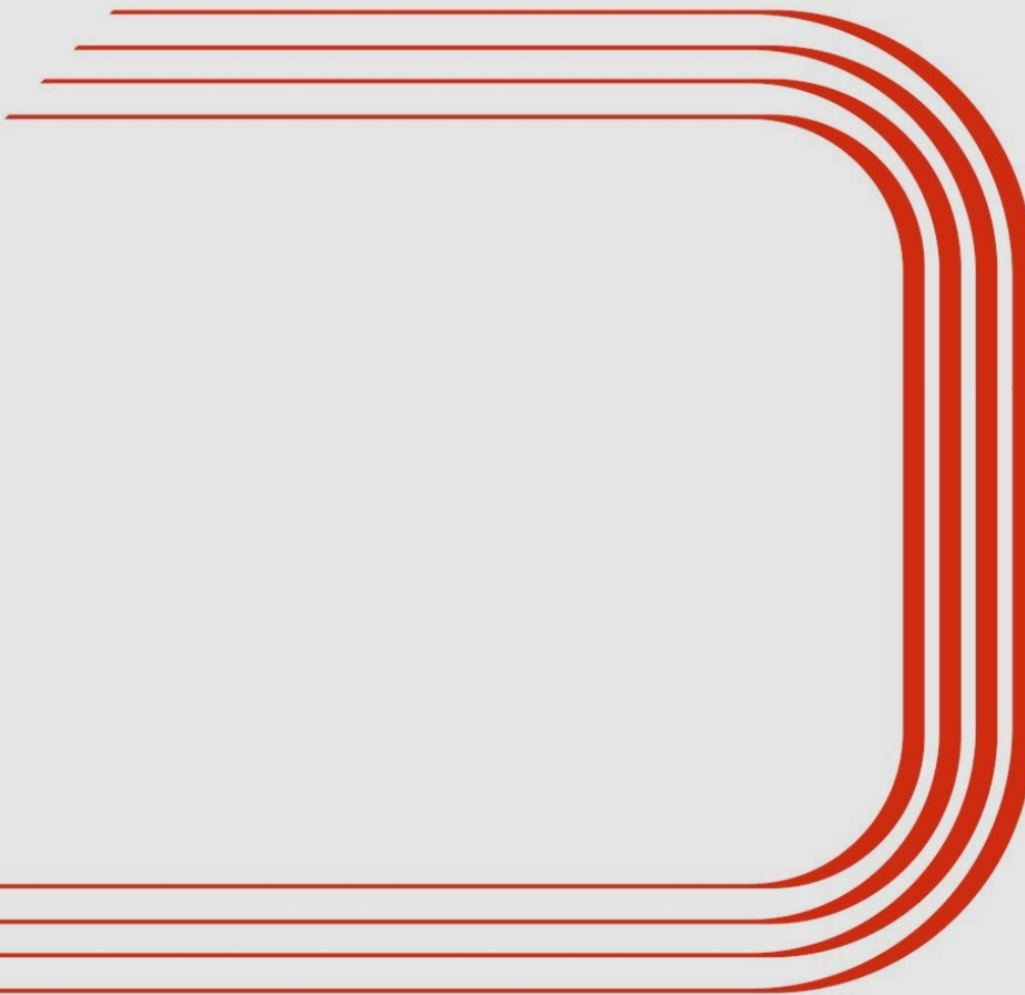


Space efficiency
Flexibility
Queryability
Consistency

JSON Relational Duality View

You Get all the Benefits of Relational Plus All the Benefits of JSON





2023



JSON Relational Duality Views

2023

ORACLE
Red Bull 
R A C I N G

S C U D E R I A

ALPHA**TAURI**

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name                varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality         varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint          pk_constructors
                      primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
```

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

New syntax to
simplify CI/CD
scripts

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name          varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality    varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint     pk_constructors
                 primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
```

if [not] exists

if [not] exists

```
SQL> drop table t;  
drop table t  
      *  
ERROR at line 1:  
ORA-00942: table or view "DEMO"."T" does not exist  
Help: https://docs.oracle.com/error-help/db/ora-00942/  
SQL> drop table if exists t;
```

Table dropped.

```
SQL>
```


if [not] exists

```
SQL> create table t
  2  ( id    number
  3    , text varchar2( 32 )
  4  )
  5  /
```

Table created.

```
SQL> create table t
  2  ( id    number
  3    , text varchar2( 32 )
  4  )
  5  /
```

```
create table t
      *
```

ERROR at line 1:

ORA-00955: name is already used by an existing object

Help: <https://docs.oracle.com/error-help/db/ora-00955/>

```
SQL>
```

if [not] exists

SQL>

if [not] exists

```
SQL> create table if not exists t
2  ( id number
3    , text varchar2( 32 )
4  )
5  /
```

Table created.

```
SQL> desc t
```

Name	Null?	Type
ID		NUMBER
TEXT		VARCHAR2 (32)

```
SQL>
```

if [not] exists

SQL>

if [not] exists

```
SQL> create table if not exists t
  2  ( id    number
  3    , text varchar2( 32 )
  4    , when date
  5  )
  6  /
```

Table created.

```
SQL> desc t
```

Name	Null?	Type
ID		NUMBER
TEXT		VARCHAR2 (32)

```
SQL>
```

No error, but nothing happened

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

New syntax to
simplify CI/CD
scripts

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name          varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality    varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint     pk_constructors
                 primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
```

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name                varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality          varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint          pk_constructors
                        primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
```

New syntax to
document your
model

Annotations

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name                varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality         varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint         pk_constructors
                    primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
SQL> select annotation_name
       2 from   user_annotations
       3 /
```

```
ANNOTATION_NAME
-----
PURPOSE
Internal key value
Name of the constructor
Nationality of the constructor
```


Annotations

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name                varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality          varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint          pk_constructors
                        primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
SQL> select annotation_name
2         , annotation_value
3   from   user_annotation_values
4   /
```

ANNOTATION_NAME	ANNOTATION_VALUE
-----	-----
PURPOSE	hold the constructors
Internal key value	
Name of the constructor	
Nationality of the constructor	

Annotations

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name                varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality         varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint         pk_constructors
                    primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
```

```
SQL> select object_name
2         , column_name
3         , annotation_name
4         , annotation_value
5 from   user_annotations_usage
6 /
```

OBJECT_NAME	COLUMN_NAME	ANNOTATION_NAME	ANNOTATION_VALUE
-----	-----	-----	-----
CONSTRUCTORS		PURPOSE	hold the constructors
CONSTRUCTORS	CONSTRUCTORID	Internal key value	
CONSTRUCTORS	NAME	Name of the constructor	
CONSTRUCTORS	NATIONALITY	Nationality of the constructor	

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, name                varchar2( 16 ) annotations ( "Name of the constructor" )
, nationality         varchar2( 16 ) annotations ( "Nationality of the constructor" )
, constraint          pk_constructors
                      primary key (constructorid)
)
annotations ( purpose 'hold the constructors' )
/
```

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```
create table if not exists constructors
(
    constructorid number          generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, driver_number number( 18 ) annotations ( "Number of the driver on the car" )
, nationality varchar2( 18 ) annotations ( "Nationality of the driver" )
, driver_name varchar2( 18 ) annotations ( "Name of the driver" )
, dob date primary key ( constructorid ) annotations ( "Birthdate of the driver" )
) nationality varchar2( 16 ) annotations ( "Nationality of the driver" )
annotations ( purpose 'hold the constructors' )
/

create table if not exists drivers
(
    driverid number          generated by default
                           on null as identity
                           annotations ( "Internal key value" )
, first_name varchar2( 18 ) annotations ( "First name of the driver" )
, last_name varchar2( 18 ) annotations ( "Last name of the driver" )
, dob date primary key ( driverid ) annotations ( "Birthdate of the driver" )
) annotations ( purpose 'hold the drivers' )
/
```

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```
create table if not exists drivers
(
    driverid          number          generated by default
                        on null as identity
                        annotations ( "Internal key value" )
, driver_number      number ( 2 ) annotations ( "Number used by driver on the car" )
, code               varchar2( 3 ) annotations ( "Callsign of the driver" )
, driver_name        varchar2( 32 ) annotations ( "Name of the driver" )
, dob                date            annotations ( "Birthdate of the driver" )
, nationality         varchar2( 16 ) annotations ( "Nationality of the driver" )
, constraint         pk_drivers
                        primary key (driverid)
)
annotations ( purpose 'hold the drivers' )
/
```

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```

create table if not exists domserveractordrivers
(
    domserveruidtordrivernumber number generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, domservernumber number(2) annotations ( "Number used by the constructor to the car" )
, domserverid varchar(2) annotations ( "Reference of the driver" )
, domservername varchar(20) annotations ( "Name of the driver" )
, dob primary key (constraint domserveruidtordriveruidtordriverdate of the driver" )
, nationality varchar(20) annotations ( "Nationality of the driver" )
, constraint preferences constructors(constructorid)
, constraint pkmainkeytoconstructorid driverid foreign key (driverid)
) references drivers(driverid)
annotations ( purpose 'hold the drivers' )
annotations ( purpose 'register which driver is connected to which constructor' )
/

```

Lets start by creating three tables, Constructors, Drivers, and the connecting table Constructordrivers

```
create table if not exists constructordrivers
(
    constructordriverid number    generated by default
                                on null as identity
                                annotations ( "Internal key value" )
, constructorid          number    annotations ( "Reference to the constructor" )
, driverid              number    annotations ( "Reference to the driver" )
, constraint             pk_constructordrivers
                        primary key (constructordriverid)
, constraint             fk_constructordrivers_constructorid foreign key (constructorid)
                        references constructors(constructorid)
, constraint             fk_constructordrivers_driverid      foreign key (driverid)
                        references drivers(driverid)
)
annotations ( purpose 'register which driver is connected to which constructor' )
/
```

Insert Constructor

```
insert into constructors
  ( name, nationality )
values
  ( 'Red Bull', 'Austrian' )
/
```


Insert Drivers

```
insert into drivers
  ( driver_number, code, driver_name, dob, nationality )
values
  ( 11, 'PER', 'Sergio Pérez' , to_date( '19900126', 'YYYYMMDD' ), 'Mexican' );
insert into drivers
  ( driver_number, code, driver_name, dob, nationality )
values
  ( 33, 'VER', 'Max Verstappen', to_date( '19970930', 'YYYYMMDD' ), 'Dutch' );
insert into drivers
  ( driver_number, code, driver_name, dob, nationality )
values
  ( 22, 'TSU', 'Yuki Tsunoda' , to_date( '20000511', 'YYYYMMDD' ), 'Japanese' );
insert into drivers
  ( driver_number, code, driver_name, dob, nationality )
values
  ( 21, 'DEV', 'Nyck de Vries' , to_date( '19950206', 'YYYYMMDD' ), 'Dutch' );
```

Insert Drivers (anonymous block)

```
begin
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 11, 'PER', 'Sergio Pérez' , to_date( '19900126', 'YYYYMMDD' ), 'Mexican' );
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 33, 'VER', 'Max Verstappen', to_date( '19970930', 'YYYYMMDD' ), 'Dutch' );
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 22, 'TSU', 'Yuki Tsunoda' , to_date( '20000511', 'YYYYMMDD' ), 'Japanese' );
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 21, 'DEV', 'Nyck de Vries' , to_date( '19950206', 'YYYYMMDD' ), 'Dutch' );
end;
/
```

Insert Drivers (anonymous block)

```
begin
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 11, 'PER', 'Sergio Pérez' , to_date( '19900126', 'YYYYMMDD' ), 'Mexican' );
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 33, 'VER', 'Max Verstappen', to_date( '19970930', 'YYYYMMDD' ), 'Dutch' );
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 22, 'TSU', 'Yuki Tsunoda' , to_date( '20000511', 'YYYYMMDD' ), 'Japanese' );
  insert into drivers
    ( driver_number, code, driver_name, dob, nationality )
  values
    ( 21, 'DEV', 'Nyck de Vries' , to_date( '19950206', 'YYYYMMDD' ), 'Dutch' );
end;
/
```

Insert Drivers (Table Values Constructor)

```
insert into drivers
  ( driver_number, code, driver_name, dob, nationality )
values
  ( 11, 'PER', 'Sergio Pérez' , to_date( '19900126', 'YYYYMMDD' ), 'Mexican' )
, ( 33, 'VER', 'Max Verstappen', to_date( '19970930', 'YYYYMMDD' ), 'Dutch' );
, ( 22, 'TSU', 'Yuki Tsunoda' , to_date( '20000511', 'YYYYMMDD' ), 'Japanese' )
, ( 21, 'DEV', 'Nyck de Vries' , to_date( '19950206', 'YYYYMMDD' ), 'Dutch' )
/
```

Insert Drivers (Table Values Constructor)

```
insert into drivers
  ( driver_number, code, driver_name, dob, nationality )
values
  ( 11, 'PER', 'Sergio Pérez' , to_date( '19900126', 'YYYYMMDD' ), 'Mexican' )
, ( 33, 'VER', 'Max Verstappen', to_date( '19970930', 'YYYYMMDD' ), 'Dutch' )
, ( 22, 'TSU', 'Yuki Tsunoda' , to_date( '20000511', 'YYYYMMDD' ), 'Japanese' )
, ( 21, 'DEV', 'Nyck de Vries' , to_date( '19950206', 'YYYYMMDD' ), 'Dutch' )
/
```

Simplified syntax, less roundtrips, faster execution

Insert ConstructorDrivers

```
insert into constructordrivers
  ( constructorid, driverid )
values
  ( ( select con.constructorid from    constructors con where  con.name = 'Red Bull' )
    , ( select drv.driverid from      drivers drv where   drv.code = 'VER' ) )
, ( ( select con.constructorid from    constructors con where  con.name = 'Red Bull' )
    , ( select drv.driverid from      drivers drv where   drv.code = 'PER' ) )
/
```

Insert ConstructorDrivers

```
insert into constructordrivers
  ( constructorid, driverid )
values
  ( ( select con.constructorid      from   constructors con where   con.name = 'Red Bull' )
    , ( select drv.driverid from     drivers drv where   drv.code = 'VER' ) )
  , ( ( select con.constructorid      from.  constructors con where   con.name = 'Red Bull' )
    , ( select drv.driverid from     drivers drv where   drv.code = 'PER' ) )
/
```

Scalar SQLMacro

```
create or replace function constructoridbyname( name_in in varchar2 )
return varchar2 sql_macro( scalar )
is
begin
    return q'[
(
    select con.constructorid
    from   constructors con
    where  con.name = name_in
)]';
end;
/
```

```
create or replace function ( code_in in varchar2 )
return varchar2 sql_macro( scalar )
is
begin
    return q'[
(
    select drv.driverid
    from   drivers drv
    where  drv.code = code_in
)]';
end driveridbycode;
/
```


Scalar SQLMacro

```
create or replace function constructoridbyname( name_in in varchar2 )
return varchar2 sql_macro( scalar )
is
begin
    return q'[
(
    select con.constructorid
    from   constructors con
    where  con.name = name_in
)]';
end;
/
```

```
create or replace function ( code_in in varchar2 )
return varchar2 sql_macro( scalar )
is
begin
    return q'[
(
    select drv.driverid
    from   drivers drv
    where  drv.code = code_in
)]';
end driveridbycode;
/
```

Insert ConstructorDrivers

```
insert into constructordrivers
  ( constructorid, driverid )
values
  ( ( select con.constructorid    from constructors con where  con.name = 'Red Bull' )
    , ( select drv.driverid      from drivers drv where   drv.code = 'VER' ) )
, ( ( select con.constructorid    from constructors con where  con.name = 'Red Bull' )
    , ( select drv.driverid      from drivers drv where   drv.code = 'PER' ) )
/
```

Insert ConstructorDrivers (using SQLMacros)

```
insert into constructordrivers
  ( constructorid, driverid )
values
  ( ( select constructorid by name for Red Bull from constructors con where con.name = 'Red Bull' )
    , ( deleteid by code for VER from drivers drv where drv.code = 'VER' ) )
, ( ( select constructorid by name for Red Bull from constructors con where con.name = 'Red Bull' )
    , ( deleteid by code for PER from drivers drv where drv.code = 'PER' ) )
/
```

Insert ConstructorDrivers (using SQLMacros)

```
insert into constructordrivers
  ( constructorid, driverid )
values
  ( ( constructoridbyname( 'Red Bull' ) )
    , ( driveridbycode( 'VER' ) ) )
, ( ( constructoridbyname( 'Red Bull' ) )
    , ( driveridbycode( 'PER' ) ) )
/
```

Select the data from the tables

```
select *  
from drivers  
/
```

DRIVERID	DRIVER_NUMBER	COD	DRIVER_NAME	DOB	NATIONALITY
1	11	PER	Sergio Perez	26/01/90	Mexican
2	33	VER	Max Verstappen	30/09/97	Dutch
3	22	TSU	Yuki Tsunoda	11/05/00	Japanese
4	21	DEV	Nyck de Vries	06/02/95	Dutch

```
select *  
from constructors  
/
```

CONSTRUCTORID	NAME	NATIONALITY
1	Red Bull	Austrian

```
select *  
from constructordrivers  
/
```

CONSTRUCTORDRIVERID	CONSTRUCTORID	DRIVERID
1	1	1
2	1	2



JSON View as it was pre-23ai

```
create or replace view constructor_v as
select json_object
  ( key '_id'          value con.constructorid
  , key 'name'         value con.name
  , key 'nationality'  value con.nationality
  , key 'drivers'      value
    ( select json_arrayagg( drivers returning clob )
      from    ( select json_object
                    ( key 'condrv' value cds.constructorid
                    , key 'driver' value json_object
                        ( key 'driverid' value drv.driverid
                        , key 'driver_number' value drv.driver_number
                        , key 'code' value drv.code
                        , key 'driver_name' value drv.driver_name
                        , key 'dob' value drv.dob
                        , key 'nationality' value drv.nationality
                        )
                    ) drivers
              from    constructordrivers cds
              join    drivers drv
              on      ( cds.driverid = drv.driverid )
              where   cds.constructorid = con.constructorid
            ) ) data
from    constructors con
```

JSON Relational Duality View – Using SQL on 23ai

```
create or replace json relational duality view constructor_dv as
select json
  ( { '_id'          :      con.constructorid
    , 'name'         :      con.name
    , 'nationality'  :      con.nationality
    , 'drivers'      :
      [ select json { 'condrv' : constructordriverid
                    , 'driver' :
                      ( select json { 'driverid'      : drv.driverid
                                    , 'driver_number' : drv.driver_number
                                    , 'code'           : drv.code
                                    , 'driver_name'    : drv.driver_name
                                    , 'dob'            : drv.dob
                                    , 'nationality'    : drv.nationality
                                    }
                                from   drivers drv with update
                                where  cds.driverid = drv.driverid
                              )
                      }
                    from   constructordrivers cds with insert update delete
                    where  con.constructorid = cds.constructorid
                  ]
      }
    )
  from constructors con with insert update delete
```


JSON Relational Duality View – Using SQL on 23ai

```
create or replace json relational duality view constructor_dv as
select json
  ( { '_id'          :      con.constructorid
    , 'name'         :      con.name
    , 'nationality'  :      con.nationality
    , 'drivers'      :
      [ select json { 'condrv' : constructordriverid
                    , 'driver' :
                      ( select json { 'driverid'      : drv.driverid
                                    , 'driver_number' : drv.driver_number
                                    , 'code'           : drv.code
                                    , 'driver_name'    : drv.driver_name
                                    , 'dob'            : drv.dob
                                    , 'nationality'    : drv.nationality
                                    }
                                from   drivers drv with update
                                where  cds.driverid = drv.driverid
                              )
                    }
          from   constructordrivers cds with insert update delete
          where  con.constructorid = cds.constructorid
        ]
      }
    )
  from   constructors con with insert update delete
```

JSON Relational Duality View – Using SQL on 23ai

```
create or replace json relational duality view constructor_dv as
select json
  ( { '_id'          :      con.constructorid
    , 'name'         :      con.name
    , 'nationality'  :      con.nationality
    , 'drivers'      :
      [ select json { 'condrv' : constructordriverid
                    , 'driver' :
                      ( select json { 'driverid'      : drv.driverid
                                    , 'driver_number' : drv.driver_number
                                    , 'code'           : drv.code
                                    , 'driver_name'    : drv.driver_name
                                    , 'dob'            : drv.dob
                                    , 'nationality'     : drv.nationality
                                    }
                                from   drivers drv with update
                                where  cds.driverid = drv.driverid
                              )
                    }
        from   constructordrivers cds with insert update delete
        where  con.constructorid = cds.constructorid
      ]
    }
  )
from   constructors con with insert update delete
```

JSON Relational Duality View – Using GraphQL 23ai

```
create or replace json relational duality view constructor_dv as
select jsons @insert @update @delete
  { '_id'      :      con.constructorid
    , 'name'    :      con.name
    , 'nationality' :      con.nationality
    , 'drivers' : constructordrivers @insert @update @delete
    [ select json { 'condrv' : constructordriverid
                  , 'driver' : drivers @update
                  ( select json { 'driverid'      : drv.driverid
                                , 'driver_number' : drv.driver_number
                                , 'code'          : drv.code
                                , 'driver_name'    : drv.driver_name
                                , 'dob'           : drv.dob
                                , 'nationality'    : drv.nationality
                                }
                  from   drivers drv with update
                  where  cds.driverid = drv.driverid
                  )
                }
        from   constructordrivers cds with insert update delete
        where  con.constructorid = cds.constructorid
      ]
    }
  from   constructors con with insert update delete
```

JSON Relational Duality View – Using GraphQL on 23ai

create or replace json relational duality view constructor_dv as
constructors @insert @update @delete

```
{  _id      :      constructorid
,  name     :      name
,  nationality :      nationality
,  drivers   : constructordrivers @insert @update @delete
  [
    {  condrid : constructordriverid
      ,  driver  : drivers @update
        {  driverid      :      driverid
          ,  driver_number :      driver_number
          ,  code         :      code
          ,  driver_name  :      driver_name
          ,  dob          :      dob
          ,  nationality  :      nationality
        }
      }
  ]
}
```

JSON Relational Duality View

```
select json_serialize( cdv.data pretty ) document
from   constructor_dv cdv
where  json_exists( cdv.data
                    , '$?(@.name like $name)'
                    passing '%Bull%' as "name"
                    )
/
```

```
{
  "_id" : 1,
  "_metadata" :
  {
    "etag" : "9C082EB91BD2D08F0F65F34E9E8E21FB",
    "asof" : "0000256758A6A0C1"
  },
  "name" : "Red Bull",
  "nationality" : "Austrian",
  "drivers" :
  [
    {
      "condrvid" : 1,
      "driver" :
      {
        "driverid" : 1,
        "driver_number" : 11,
        "code" : "PER",
        "driver_name" : "Sergio Perez",
        "dob" : "1990-01-26T00:00:00",
        "nationality" : "Mexican"
      }
    },
    {
      "condrvid" : 2,
      "driver" :
      {
        "driverid" : 2,
        "driver_number" : 33,
        "code" : "VER",
        "driver_name" : "Max Verstappen",
        "dob" : "1997-09-30T00:00:00",
        "nationality" : "Dutch"
      }
    }
  ]
}
```

JSON Relational Duality View

```
select json_serialize( cdv.data pretty )
from   constructor_dv cdv
where  json_exists( cdv.data
                    , '$?(@.name like
                        passing '%Bull
                    )
/
```

```
update drivers
set driver_number = 1
where driver_number = 33
/
commit
/
```

```
{
  "_id" : 1,
  "_metadata" :
  {
    "etag" : "9C082EB91BD2D08F0F65F34E9E8E21FB",
    "asof" : "0000256758A6A0C1"
  },
  "name" : "Red Bull",
  "nationality" : "Austrian",
  "drivers" :
  {
    "condrvid" : 1,
    "driver" :
    {
      "driverid" : 1,
      "driver_number" : 11,
      "code" : "PER",
      "driver_name" : "Sergio Perez",
      "dob" : "1990-01-26T00:00:00",
      "nationality" : "Mexican"
    },
    {
      "condrvid" : 2,
      "driver" :
      {
        "driverid" : 2,
        "driver_number" : 33,
        "code" : "VER",
        "driver_name" : "Max Verstappen",
        "dob" : "1997-09-30T00:00:00",
        "nationality" : "Dutch"
      }
    }
  }
}
```

JSON Relational Duality View

```
select json_serialize( cdv.data pretty ) document
from   constructor_dv cdv
where  json_exists( cdv.data
                    , '$?(@.name like $name)'
                    passing '%Bull%' as "name"
                    )
/
```

```
{
  "_id" : 1,
  "_metadata" :
  {
    "etag" : "9C082EB91BD2D08F0F65F34E9E8E21FB",
    "asof" : "0000256758A6A0C1"
  },
  "name" : "Red Bull",
  "nationality" : "Austrian",
  "drivers" :
  [
    {
      "condrvid" : 1,
      "driver" :
      {
        "driverid" : 1,
        "driver_number" : 11,
        "code" : "PER",
        "driver_name" : "Sergio Perez",
        "dob" : "1990-01-26T00:00:00",
        "nationality" : "Mexican"
      }
    },
    {
      "condrvid" : 2,
      "driver" :
      {
        "driverid" : 2,
        "driver_number" : 1,
        "code" : "VER",
        "driver_name" : "Max Verstappen",
        "dob" : "1997-09-30T00:00:00",
        "nationality" : "Dutch"
      }
    }
  ]
}
```

Insert into Duality View

```
insert into constructor_dv
values ( '
{ "name" : "AlphaTauri"
, "nationality" : "Italian"
, "drivers" :
  [ { "driver" :
      { "driverid" : 3
        , "driver_number" : 22
        , "code" : "TSU"
        , "driver_name" : "Yuki Tsunoda"
        , "dob" : "2000-05-11T00:00:00"
        , "nationality" : "Japanese"
      }
    }
  , { "driver" :
      { "driverid" : 4
        , "driver_number" : 21
        , "code" : "DEV"
        , "driver_name" : "Nyck de Vries"
        , "dob" : "1995-02-06T00:00:00"
        , "nationality" : "Dutch"
      }
    }
  ]
}
' )
/
```


Read inserted data using Duality View

```
{
  "_id" : 2,
  "_metadata" :
  {
    "etag" : "70784C8AA93753953E859D221B160177",
    "asof" : "0000256758A6AFA3"
  },
  "name" : "AlphaTauri",
  "nationality" : "Italian",
  "drivers" :
  [
    {
      "condrvid" : 3,
      "driver" :
      {
        "driverid" : 4,
        "driver_number" : 21,
        "code" : "DEV",
        "driver_name" : "Nyck de Vries",
        "dob" : "1995-02-06T00:00:00",
        "nationality" : "Dutch"
      }
    },
    {
      "condrvid" : 4,
      "driver" :
      {
        "driverid" : 3,
        "driver_number" : 22,
        "code" : "TSU",
        "driver_name" : "Yuki Tsunoda",
        "dob" : "2000-05-11T00:00:00",
        "nationality" : "Japanese"
      }
    }
  ]
}
```

Underlying table data has changed after the Update and Insert

```
select *  
from drivers  
/
```

DRIVERID	DRIVER_NUMBER	COD	DRIVER_NAME	DOB	NATIONALITY
1	11	PER	Sergio Perez	26/01/90	Mexican
2	1	VER	Max Verstappen	30/09/97	Dutch
3	22	TSU	Yuki Tsunoda	11/05/00	Japanese
4	21	DEV	Nyck de Vries	06/02/95	Dutch

```
select *  
from constructors  
/
```

CONSTRUCTORID	NAME	NATIONALITY
1	Red Bull	Austrian
2	AlphaTauri	Italian

```
select *  
from constructordrivers  
/
```

CONSTRUCTORDRIVERID	CONSTRUCTORID	DRIVERID
1	1	1
2	1	2
3	2	4
4	2	3

JSON Relational Duality View using GraphQL

```
create or replace json relational duality view driver_dv as
drivers @insert @update @delete
{ _id          : driverid
, driver_number : driver_number
, code         : code
, driver_name   : driver_name
, dob          : dob
, nationality    : nationality
, constructor    : constructordrivers @insert @update @delete
  { constructordriverid : constructordriverid
    constructors @noinsert @update @nodelete @unnest
    { constructorid : constructorid
      , name         : name
      , nationality   : nationality
    }
  }
}
```

JSON Relational Duality View using GraphQL - unnest

```
create or replace json relational duality view driver_dv as
drivers @insert @update @delete
{ _id          : driverid
, driver_number : driver_number
, code         : code
, driver_name  : driver_name
, dob         : dob
, nationality   : nationality
, constructor   : constructordrivers @insert @update @delete
  { constructordriverid : constructordriverid
    constructors @noinsert @update @nodelete @unnest
    { constructorid : constructorid
      , name        : name
      , nationality  : nationality
    }
  }
}
```

Duality View - metadata - etag

```
select json_serialize( cdv.data pretty )
from   driver_dv cdv
where  json_exists( cdv.data
                   , '$?(@.driver_name like $name)'
                   , passing '%Verstappen%' as "name"
                   )
```

We need
the _metadata
tag when we
update the
document

```
{ "_id" : 2,
  "_metadata" :
  {
    "etag" : "0B1DA6C654D04244E9DB4BBBE8D16CFD",
    "asof" : "00002567DFD9BA9C"
  },
  "driver_number" : 33,
  "code" : "VER",
  "driver_name" : "Max Verstappen",
  "dob" : "1997-09-30T00:00:00",
  "nationality" : "Dutch",
  "constructor" :
  [
    {
      "constructordriverid" : 1,
      "constructorid" : 1,
      "name" : "Red Bull",
      "nationality" : "Austrian"
    }
  ]
}
```

Duality View - metadata - etag

```
update driver_dv ddv
set   data = '{ "_id" : 2,
                "_metadata" :
                {
                  "etag" : "0B1DA6C654D04244E9DB4BBBE8D16CFD",
                  "asof" : "00002567DFD9BA9C"
                },
                "driver_number" : 33,
                "code" : "VER",
                "driver_name" : "Max Verstappen",
                "dob" : "1997-09-30T00:00:00",
                "nationality" : "Dutch",
                "constructor" :
                [
                  {
                    "constructordriverid" : 1,
                    "constructorid" : 1,
                    "name" : "Red Bull",
                    "nationality" : "Austrian"
                  }
                ]
              }'
where json_value ( ddv.data, '$._id' ) = 2
/
```

We need
the _metadata
tag when we
update the
document

Duality View - metadata - etag

```
update driver_dv ddv
set   data = '{ "_id" : 2,
                 "_metadata" :
                 {
                   "etag" : "0B1DA6C654D04244E9DB4BBBE8D16CFD",
                   "asof" : "00002567DFD9BA9C"
                 },
                 "driver_number" : 33,
                 "code" : "VER",
                 "driver_name" : "Max Verstappen",
                 "dob" : "1997-09-30T00:00:00",
                 "nationality" : "Dutch",
                 "constructor" :
                 [
                   {
                     "constructordriverid" : 1,
                     "constructorid" : 1,
                     "name" : "Red Bull",
                     "nationality" : "Austrian"
                   }
                 ]
               }'
where json_value ( ddv.data, '$._id' ) = 2
/
```

We need
the _metadata
tag when we
update the
document

Duality View - metadata - etag

```
update driver_dv ddv
set   data = '{ "_id" : 2,
                "_metadata" :
                {
                  "etag" : "0B1DA6C654D04244E9DB4BBBE8D16CFD",
                  "asof" : "00002567DFD9BA9C"
                },
                "driver_number" : 33,
                "code" : "VER",
                "driver_name" : "Max Verstappen",
                "dob" : "1997-09-30T00:00:00",
                "nationality" : "Dutch",
                "constructor" :
                [
                  {
                    "constructordriverid" : 1,
                    "constructorid" : 1,
                    "name" : "Oracle Red Bull",
                    "nationality" : "Austrian"
                  }
                ]
              }'
where json_value ( ddv.data, '$._id' ) = 2
/
```

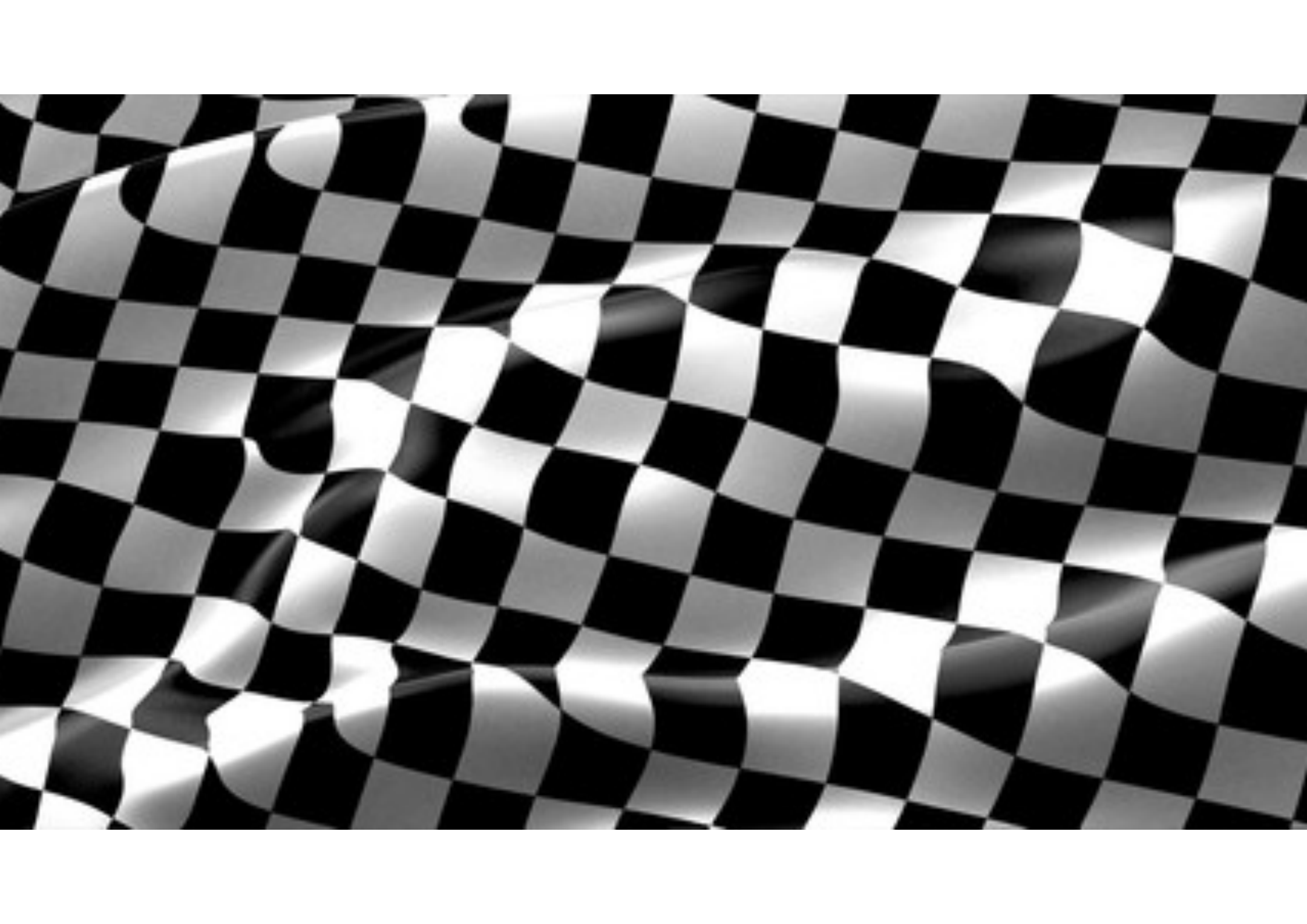
We need
the _metadata
tag when we
update the
document

Duality View – All JSON documents get updated

```
select json_serialize( cdv.data pretty )
from   driver_dv cdv
where  json_exists( cdv.data
                    , '$?(@.constructor.name like $name)'
                    passing '%Bull%' as "name"
                    )
```

```
{
  "_id" : 1,
  "_metadata" :
  {
    "etag" : "B31440991D34177EE1340A6BE3576366",
    "asof" : "00002567DFD9BF30"
  },
  "driver_number" : 11,
  "code" : "PER",
  "driver_name" : "Sergio Perez",
  "dob" : "1990-01-26T00:00:00",
  "nationality" : "Mexican",
  "constructor" :
  [
    {
      "constructordriverid" : 2,
      "constructorid" : 1,
      "name" : "Oracle Red Bull",
      "nationality" : "Austrian"
    }
  ]
}
```

```
{
  "_id" : 2,
  "_metadata" :
  {
    "etag" : "C4BA6A4B3BA8FE4C655B154C5FC68C43",
    "asof" : "00002567DFD9BF30"
  },
  "driver_number" : 33,
  "code" : "VER",
  "driver_name" : "Max Verstappen",
  "dob" : "1997-09-30T00:00:00",
  "nationality" : "Dutch",
  "constructor" :
  [
    {
      "constructordriverid" : 1,
      "constructorid" : 1,
      "name" : "Oracle Red Bull",
      "nationality" : "Austrian"
    }
  ]
}
```

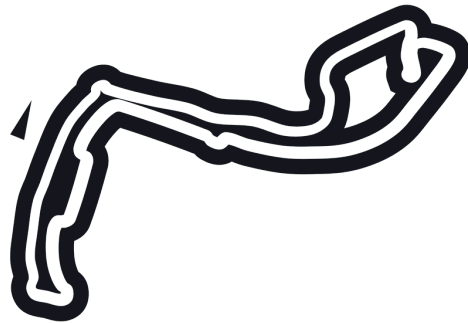
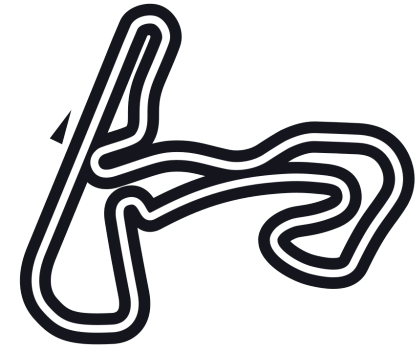


<http://ergast.com/api/f1/circuits.json?limit=1000>

<http://ergast.com/api/f1/circuits.json?limit=1000>

```
{ "lat": "25.7617", "long": "-80.1917", "locality": "Miami", "country": "USA" } },
{ "circuitId": "monaco", "url": "http://en.wikipedia.org/wiki/Circuit_de_Monaco", "circuitName": "Circuit de Monaco", "Location":
{ "lat": "43.7347", "long": "7.42056", "locality": "Monte-Carlo", "country": "Monaco" } },
{ "circuitId": "monsanto", "url": "http://en.wikipedia.org/wiki/Monsanto_Park_Circuit", "circuitName": "Monsanto Park Circuit", "Location":
{ "lat": "38.7197", "long": "-9.20306", "locality": "Lisbon", "country": "Portugal" } },
{ "circuitId": "montjuic", "url": "http://en.wikipedia.org/wiki/Montju%C3%AFC_circuit", "circuitName": "Montjuïc", "Location":
{ "lat": "41.3664", "long": "2.15167", "locality": "Barcelona", "country": "Spain" } },
{ "circuitId": "monza", "url": "http://en.wikipedia.org/wiki/Autodromo_Nazionale_Monza", "circuitName": "Autodromo Nazionale di Monza", "Location":
{ "lat": "45.6156", "long": "9.28111", "locality": "Monza", "country": "Italy" } },
{ "circuitId": "mosport", "url": "http://en.wikipedia.org/wiki/Mosport", "circuitName": "Mosport International Raceway", "Location":
{ "lat": "44.0481", "long": "-78.6756", "locality": "Ontario", "country": "Canada" } },
{ "circuitId": "mugello", "url": "http://en.wikipedia.org/wiki/Mugello_Circuit", "circuitName": "Autodromo Internazionale del Mugello", "Location":
{ "lat": "43.9975", "long": "11.3719", "locality": "Mugello", "country": "Italy" } },
{ "circuitId": "nivelles", "url": "http://en.wikipedia.org/wiki/Nivelles-Grand-Prix-Circuit", "circuitName": "Nivelles Grand Prix Circuit", "Location":
{ "lat": "50.8278", "long": "4.83583", "locality": "Nivelles", "country": "Belgium" } },
```

```
{  
  "circuitId": "zandvoort",  
  "url": "http://en.wikipedia.org/wiki/Circuit_Zandvoort",  
  "circuitName": "Circuit Park Zandvoort",  
  "Location": {  
    "lat": "52.3888",  
    "long": "4.54092",  
    "locality": "Zandvoort",  
    "country": "Netherlands"  
  }  
}
```



```
{  
  "circuitId": "monaco",  
  "url": "http://en.wikipedia.org/wiki/Circuit_de_Monaco",  
  "circuitName": "Circuit de Monaco",  
  "Location": {  
    "lat": "43.7347",  
    "long": "7.42056",  
    "locality": "Monte-Carlo",  
    "country": "Monaco"  
  }  
}
```

Create table and load documents

```
create table if not exists tracks( data json)
/  
  
create table if not exists seasons( data json)
/  
  
create table if not exists races( data json)
/
```

Create table and load documents

```
insert into seasons
( data )
values
( '{"season":"1950","url":"http:\\\\en.wikipedia.org/wiki/1950_Formula_One_season"}' )
, ( '{"season":"1951","url":"http:\\\\en.wikipedia.org/wiki/1951_Formula_One_season"}' )
, ( '{"season":"1952","url":"http:\\\\en.wikipedia.org/wiki/1952_Formula_One_season"}' )
, ( '{"season":"1953","url":"http:\\\\en.wikipedia.org/wiki/1953_Formula_One_season"}' )
, ( '{"season":"1954","url":"http:\\\\en.wikipedia.org/wiki/1954_Formula_One_season"}' )
, ( '{"season":"1955","url":"http:\\\\en.wikipedia.org/wiki/1955_Formula_One_season"}' )
, ( '{"season":"1956","url":"http:\\\\en.wikipedia.org/wiki/1956_Formula_One_season"}' )
, ( '{"season":"1957","url":"http:\\\\en.wikipedia.org/wiki/1957_Formula_One_season"}' )
...
, ( '{"season":"2019","url":"http:\\\\en.wikipedia.org/wiki/2019_Formula_One_World_Championship"}' )
, ( '{"season":"2020","url":"http:\\\\en.wikipedia.org/wiki/2020_Formula_One_World_Championship"}' )
, ( '{"season":"2021","url":"http:\\\\en.wikipedia.org/wiki/2021_Formula_One_World_Championship"}' )
, ( '{"season":"2022","url":"http:\\\\en.wikipedia.org/wiki/2022_Formula_One_World_Championship"}' )
, ( '{"season":"2023","url":"http:\\\\en.wikipedia.org/wiki/2023_Formula_One_World_Championship"}' )
, ( '{"season":"2024","url":"https:\\\\en.wikipedia.org/wiki/2024_Formula_One_World_Championship"}' )
/
```

Create table and load documents

```
insert into tracks
( data )
values
( '{"circuitId":"adelaide","url":"http://en.wikipedia.org/wiki/Adelaide_Street_Circuit","circuitName":"Adelaide Street Circuit","Location":{"lat":"-34.9272","long":"138.617","locality":"Adelaide","country":"Australia"}}' )
, ( '{"circuitId":"ain-diab","url":"http://en.wikipedia.org/wiki/Ain-Diab_Circuit","circuitName":"Ain Diab","Location":{"lat":"33.5786","long":"-7.6875","locality":"Casablanca","country":"Morocco"}}' )
, ...
( '{"circuitId":"zandvoort","url":"http://en.wikipedia.org/wiki/Circuit_Zandvoort","circuitName":"Circuit Park Zandvoort","Location":{"lat":"52.3888","long":"4.54092","locality":"Zandvoort","country":"Netherlands"}}' )
, ( '{"circuitId":"zeltweg","url":"http://en.wikipedia.org/wiki/Zeltweg_Airfield","circuitName":"Zeltweg","Location":{"lat":"47.2039","long":"14.7478","locality":"Styria","country":"Austria"}}' )
, ( '{"circuitId":"zolder","url":"http://en.wikipedia.org/wiki/Zolder","circuitName":"Zolder","Location":{"lat":"50.9894","long":"5.25694","locality":"Heusden-Zolder","country":"Belgium"}}' )
/
```


Create table and load documents

```
insert into races
( data )
values
( ' {"season":"2023","round":"1","url":"https://en.wikipedia.org/wiki/
2023_Bahrain_Grand_Prix","raceName":"Bahrain Grand Prix","Circuit":
{"circuitId":"bahrain","url":"http://en.wikipedia.org/wiki/
Bahrain_International_Circuit","circuitName":"Bahrain International Circuit","Location":
{"lat":"26.0325","long":"50.5106","locality":"Sakhir","country":"Bahrain"}}, "date":"2023-03-05","time
":"15:00:00Z","Qualifying":{"date":"2023-03-04","time":"15:00:00Z"}}' )
, ( ' {"season":"2023","round":"2","url":"https://en.wikipedia.org/wiki/
2023_Saudi_Arabian_Grand_Prix","raceName":"Saudi Arabian Grand Prix","Circuit":
{"circuitId":"jeddah","url":"http://en.wikipedia.org/wiki/
Jeddah_Street_Circuit","circuitName":"Jeddah Corniche Circuit","Location":
{"lat":"21.6319","long":"39.1044","locality":"Jeddah","country":"Saudi
Arabia"}}, "date":"2023-03-19","time":"17:00:00Z","Qualifying":
{"date":"2023-03-18","time":"17:00:00Z"}}' )
...
, ( ' {"season":"2023","round":"22","url":"https://en.wikipedia.org/wiki/
2023_Abu_Dhabi_Grand_Prix","raceName":"Abu Dhabi Grand Prix","Circuit":
{"circuitId":"yas_marina","url":"http://en.wikipedia.org/wiki/
Yas_Marina_Circuit","circuitName":"Yas Marina Circuit","Location":
{"lat":"24.4672","long":"54.6031","locality":"Abu
Dhabi","country":"UAE"}}, "date":"2023-11-26","time":"13:00:00Z","Qualifying":
{"date":"2023-11-25","time":"14:00:00Z"}}' )
/
```

```

declare
    schema_sql clob;
begin
    schema_sql :=
        dbms_json_duality.infer_and_generate_schema
        ( json( '{ "outputFormat"   : "executable"
                    , "sourceSchema"  : "DEMO"
                    , "tableNames"    : [ "SEASONS", "TRACKS", "RACES" ]
                    , "targetSchema"  : "DEMO"
                    , "updatability"   : true
                    , "useFlexFields"  : false
                    , "viewNames"     : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
                }'
            )
        );
    dbms_output.put_line( 'ddl script: ' );
    dbms_output.put_line( schema_sql );
    execute immediate schema_sql;
    dbms_json_duality.import( table_name => 'TRACKS'
                             , view_name  => 'TRACKS_DV' );
    dbms_json_duality.import( table_name => 'SEASONS'
                             , view_name  => 'SEASONS_DV' );
    dbms_json_duality.import( table_name => 'RACES'
                             , view_name  => 'RACES_DV' );
end;

```

```

declare
    schema_sql clob;
begin
    schema_sql :=
        dbms_json_duality.infer_and_generate_schema
        ( json( '{ "outputFormat"   : "executable"
                    , "sourceSchema"  : "DEMO"
                    , "tableNames"    : [ "SEASONS", "TRACKS", "RACES" ]
                    , "targetSchema"  : "DEMO"
                    , "updatability"  : true
                    , "useFlexFields" : false
                    , "viewNames"     : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
                  }'
          )
        );
    dbms_output.put_line( 'ddl script: ' );
    dbms_output.put_line( schema_sql );
    execute immediate schema_sql;
    dbms_json_duality.import( table_name => 'TRACKS'
                             , view_name  => 'TRACKS_DV' );
    dbms_json_duality.import( table_name => 'SEASONS'
                             , view_name  => 'SEASONS_DV' );
    dbms_json_duality.import( table_name => 'RACES'
                             , view_name  => 'RACES_DV' );
end;

```

```

declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema
    ( json( '{ "outputFormat" : "executable"
              , "sourceSchema" : "DEMO"
              , "tableNames"   : [ "SEASONS", "TRACKS", "RACES" ]
              , "targetSchema" : "DEMO"
              , "updatability"  : true
              , "useFlexFields" : false
              , "viewNames"    : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
            }'
        )
    );
  dbms_output.put_line( 'ddl script: ' );
  dbms_output.put_line( schema_sql );
  execute immediate schema_sql;
  dbms_json_duality.import( table_name => 'TRACKS'
                           , view_name  => 'TRACKS_DV' );
  dbms_json_duality.import( table_name => 'SEASONS'
                           , view_name  => 'SEASONS_DV' );
  dbms_json_duality.import( table_name => 'RACES'
                           , view_name  => 'RACES_DV' );
end;

```

Create an executable script

Tables where JSON is stored

Duality views to be created

```

declare
    schema_sql clob;
begin
    schema_sql :=
        dbms_json_duality.infer_and_generate_schema
        ( json( '{ "outputFormat"   : "executable"
                  , "sourceSchema"   : "DEMO"
                  , "tableNames"     : [ "SEASONS", "TRACKS", "RACES" ]
                  , "targetSchema"   : "DEMO"
                  , "updatability"    : true
                  , "useFlexFields"   : false
                  , "viewNames"      : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
                }'
        )
    );
    dbms_output.put_line( 'ddl script: ' );
    dbms_output.put_line( schema_sql );
    execute immediate schema_sql;
    dbms_json_duality.import( table_name => 'TRACKS'
                             , view_name  => 'TRACKS_DV' );
    dbms_json_duality.import( table_name => 'SEASONS'
                             , view_name  => 'SEASONS_DV' );
    dbms_json_duality.import( table_name => 'RACES'
                             , view_name  => 'RACES_DV' );
end;

```

Execute the
script and create
the objects

```

declare
    schema_sql clob;
begin
    schema_sql :=
        dbms_json_duality.infer_and_generate_schema
        ( json( '{ "outputFormat" : "executable"
                    , "sourceSchema" : "DEMO"
                    , "tableNames" : [ "SEASONS", "TRACKS", "RACES" ]
                    , "targetSchema" : "DEMO"
                    , "updatability" : true
                    , "useFlexFields" : false
                    , "viewNames" : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
                }'
            )
        );
    dbms_output.put_line( 'ddl script: ' );
    dbms_output.put_line( schema_sql );
    execute immediate schema_sql;
    dbms_json_duality.import( table_name => 'TRACKS'
                             , view_name => 'TRACKS_DV' );
    dbms_json_duality.import( table_name => 'SEASONS'
                             , view_name => 'SEASONS_DV' );
    dbms_json_duality.import( table_name => 'RACES'
                             , view_name => 'RACES_DV' );
end;

```

Table where
JSON is stored

Duality view
that has been
created

```

declare
    schema_sql clob;
begin
    schema_sql :=
        dbms_json_duality.infer_and_generate_schema
        ( json( '{ "outputFormat" : "executable"
                    , "sourceSchema" : "DEMO"
                    , "tableNames" : [ "SEASONS", "TRACKS", "RACES" ]
                    , "targetSchema" : "DEMO"
                    , "updatability" : true
                    , "useFlexFields" : false
                    , "viewNames" : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
                }'
            )
        );
    dbms_output.put_line( 'ddl script: ' );
    dbms_output.put_line( schema_sql );
    execute immediate schema_sql;
    dbms_json_duality.import( table_name => 'TRACKS'
                             , view_name => 'TRACKS_DV' );
    dbms_json_duality.import( table_name => 'SEASONS'
                             , view_name => 'SEASONS_DV' );
    dbms_json_duality.import( table_name => 'RACES'
                             , view_name => 'RACES_DV' );
end;

```

Table where
JSON is stored

Duality view
that has been
created

```

declare
    schema_sql clob;
begin
    schema_sql :=
        dbms_json_duality.infer_and_generate_schema
        ( json( '{ "outputFormat" : "executable"
                  , "sourceSchema" : "DEMO"
                  , "tableNames"   : [ "SEASONS", "TRACKS", "RACES" ]
                  , "targetSchema" : "DEMO"
                  , "updatability" : true
                  , "useFlexFields" : false
                  , "viewNames"    : [ "SEASONS_DV", "TRACKS_DV", "RACES_DV" ]
                }'
        )
    );
    dbms_output.put_line( 'ddl script: ' );
    dbms_output.put_line( schema_sql );
    execute immediate schema_sql;
    dbms_json_duality.import( table_name => 'TRACKS'
                             , view_name  => 'TRACKS_DV' );
    dbms_json_duality.import( table_name => 'SEASONS'
                             , view_name  => 'SEASONS_DV' );
    dbms_json_duality.import( table_name => 'RACES'
                             , view_name  => 'RACES_DV' );
end;

```

Table where
JSON is stored

Duality view
that has been
created

Results in the created duality view

```
select json_serialize( tdv.data pretty ) document
from   tracks_dv tdv
where  json_exists( tdv.data
                    , '$?(@.circuitId like $name)'
                    passing '%zandvoort%' as "name"
                    )
      {
        "_id" : "zandvoort",
        "_metadata" :
        {
          "etag" : "E54F3F3888AF9B1AC745DDCC4AA3A08C",
          "asof" : "00000000005CE480"
        },
        "url" : "http://en.wikipedia.org/wiki/Circuit_Zandvoort",
        "Location" :
        {
          "lat" : "52.3888",
          "long" : "4.54092",
          "country" : "Netherlands",
          "locality" : "Zandvoort"
        },
        "circuitName" : "Circuit Park Zandvoort",
        "circuitId" : "zandvoort"
      }
```

Results in the created relational tables

```
select *
from   tracks_dv_root tdr
where  tdr.circuit_id = 'zandvoort'
/
```

URL	CIRCUIT_ID	CIRCUIT_NAME	LAT_TRACKS_DV_LOCATION
http://en.wikipedia.org/wiki/Circuit_Zandvoort	zandvoort	Circuit Park Zandvoort	52.3888

```
select *
from   tracks_dv_location tdl
where  tdl.lat = 52.3888
/
```

LAT	ORA\$LONG	COUNTRY	LOCALITY
52.3888	4.54092	Netherlands	Zandvoort

JSON Duality

Delivers development simplicity of JSON + the scalability of relational

Eliminates data duplication problem from document model

Minimizes database round trips

Data can be read and written as documents or as tables

Build document-centric apps on existing relational data

Build relational schemas on existing documents

Q&A

Please fill in your
evaluations

