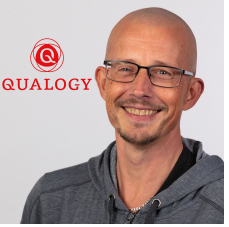
How to use a relational database for your JSON documents



About me

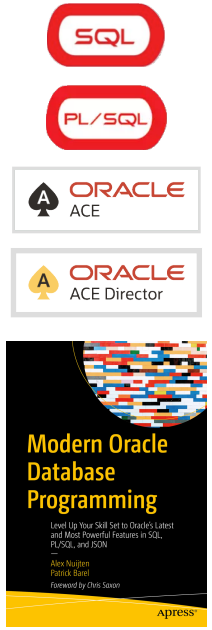
Experience since 1997

Discovered PL/SQL in 1999

Oracle ACE since 2011

Oracle ACE Director since 2019

Modern Oracle Database Programming 2023



Patrick Barel

PBarel@Qualogy.com

Patrick.Barel@GMail.com

patrick@bar-solutions.com

@patch72

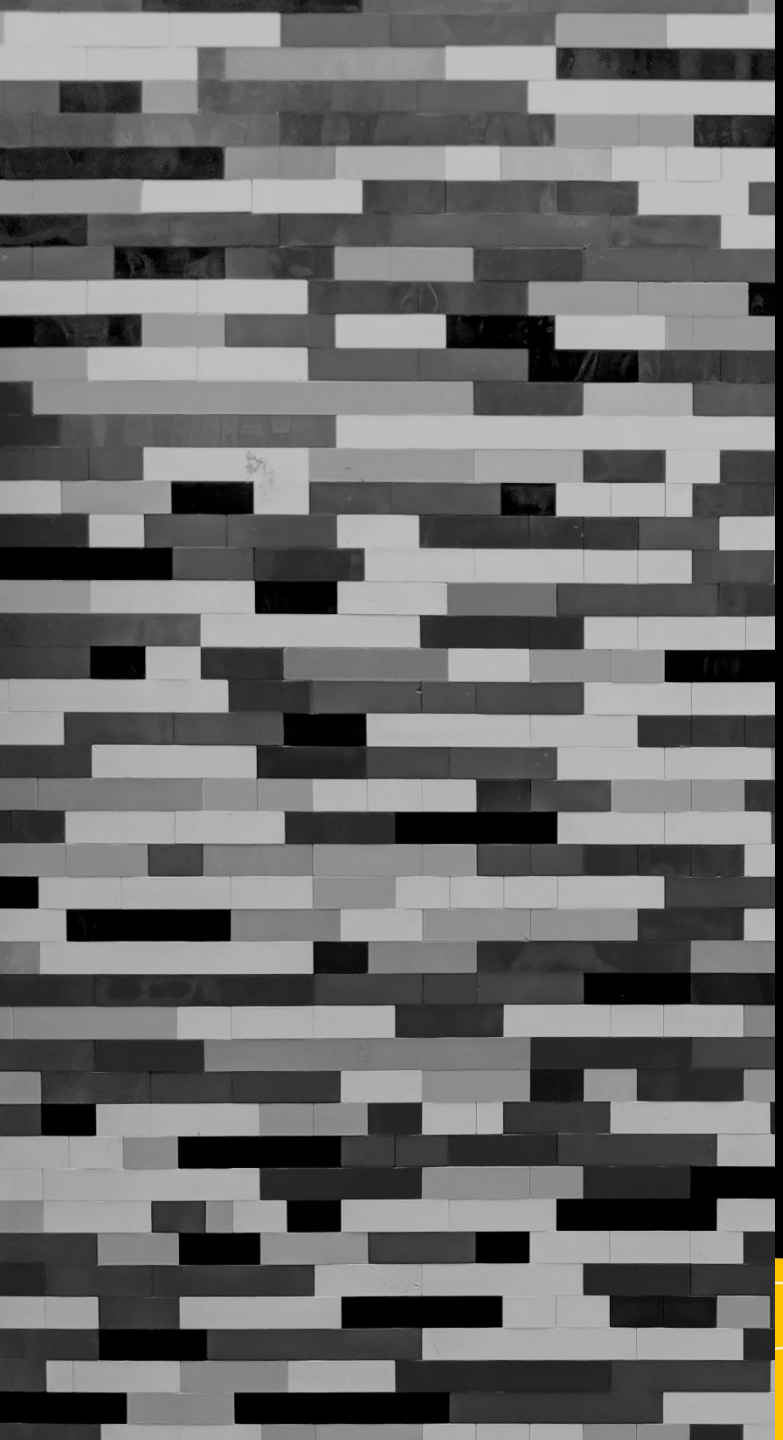
bar-solutions.com blog http://blog.bar-solutions.com

~~All things ORACLE http://allthingsoracle.com~~

~~OTECH MAGAZINE http://www.otechmag.com~~

www.red-gate.com/simple-talk/author/patrick-barel/

bar-solutions.com/otechmagazine.php



Modern Oracle Database Programming

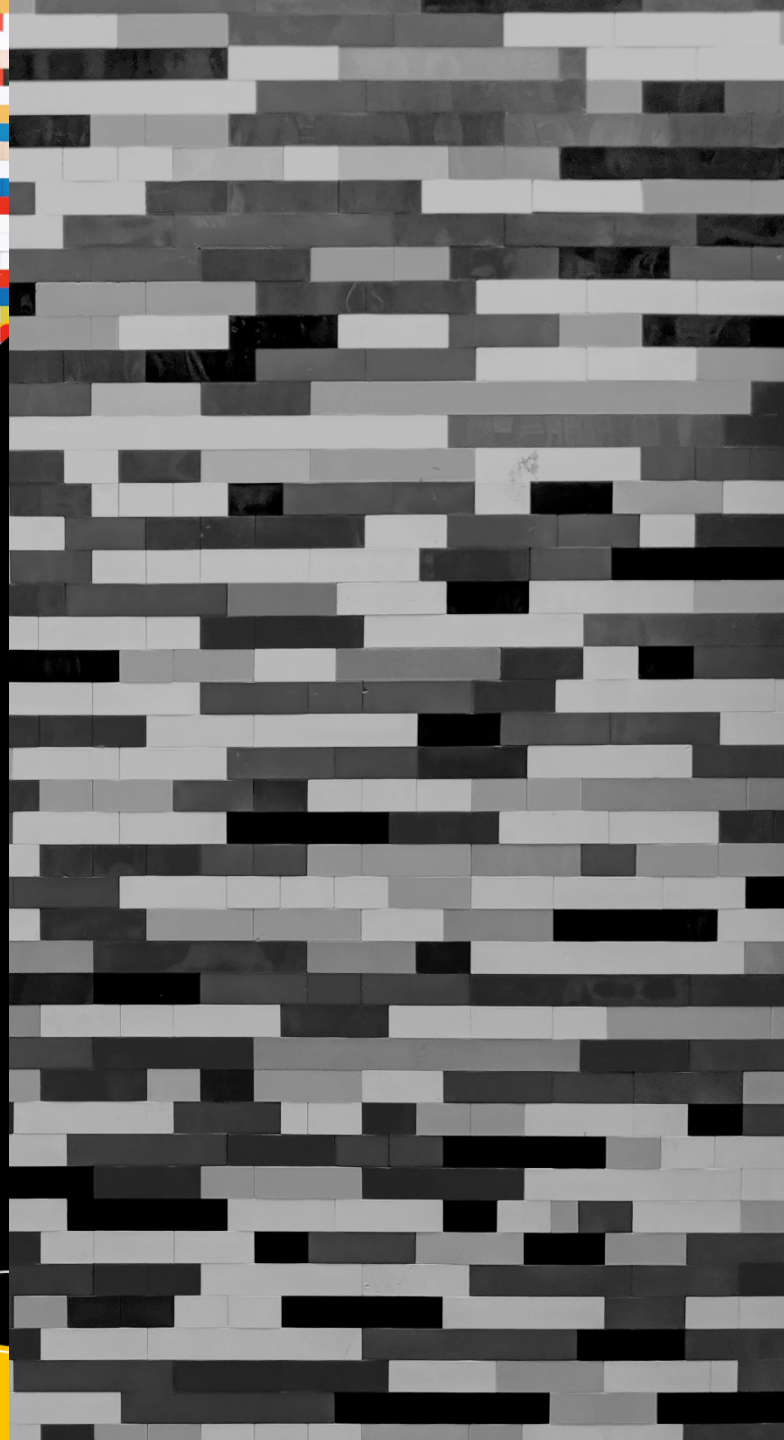
Level Up Your Skill Set to Oracle's Latest and Most Powerful Features in SQL, PL/SQL, and JSON

—

Alex Nuijten
Patrick Barel

Foreword by Chris Saxon

Apress®



Oracle ACE

Tech Superstars Unite

Get worldwide recognition as an Oracle ACE

Oracle.com profile page

Swag, certification exam credit & event passes

Exclusive content

Networking events

Your own Oracle cloud account

Travel support

@oracleace

Linkedin.com/groups/72183

@oracleace.bsky.social

Learn more at:
ace.oracle.com

Oracle Cloud Infrastructure

New Free Tier

oracle.com/cloud/free

https://patch72.com/keepalive

Always Free

Services you can use for unlimited time

+

30-Day Free Trial

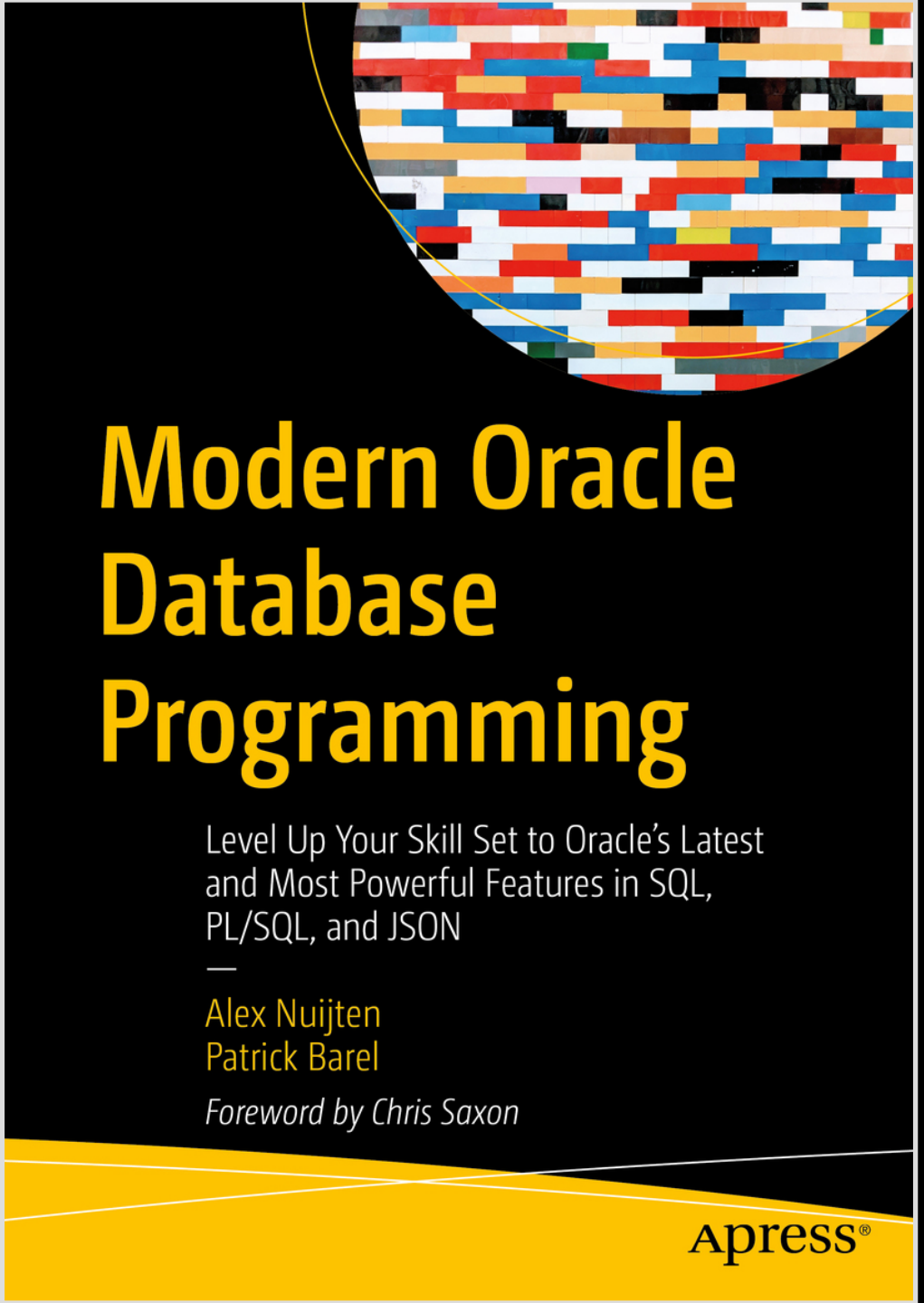
Free credits you can use for more services

How to use a relational database for your JSON documents

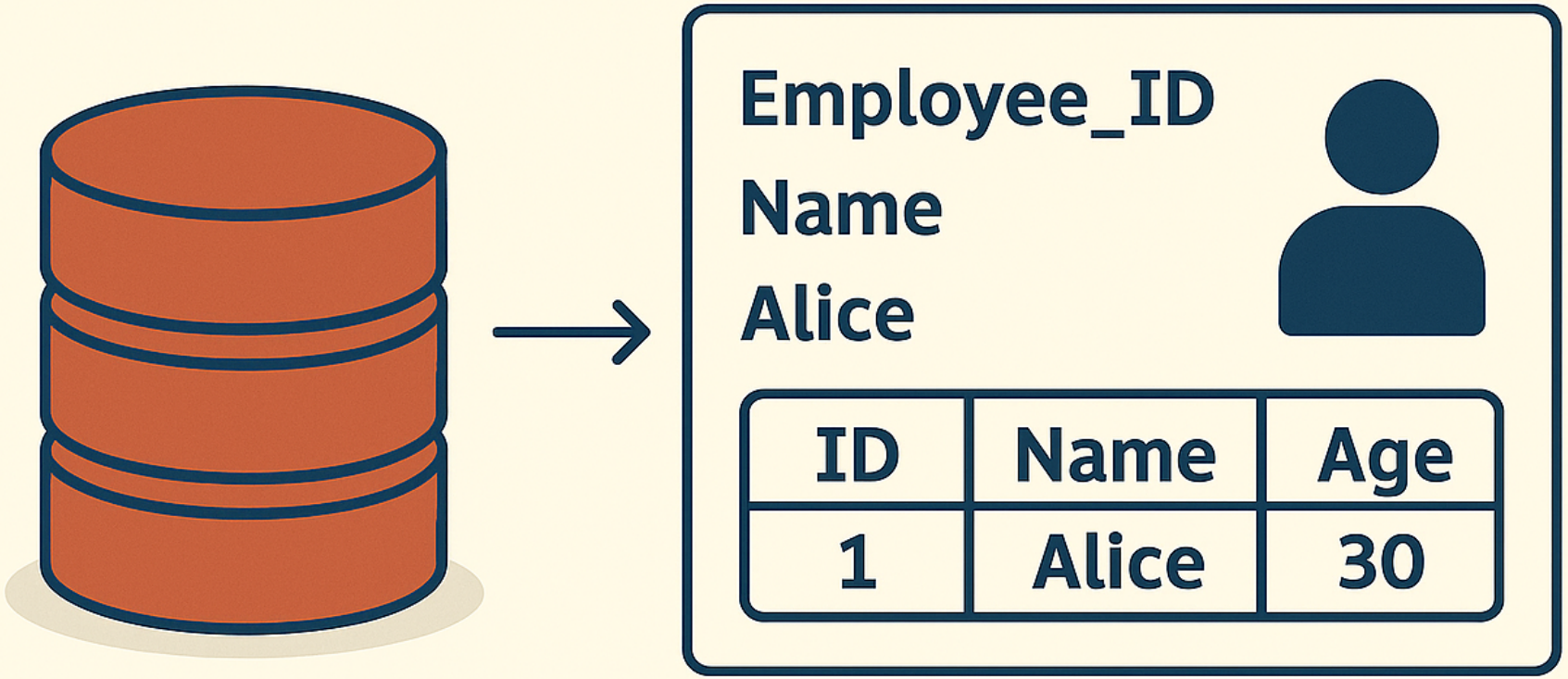
Patrick Barel, Qualogy

November 13, 2025

QUALOGY

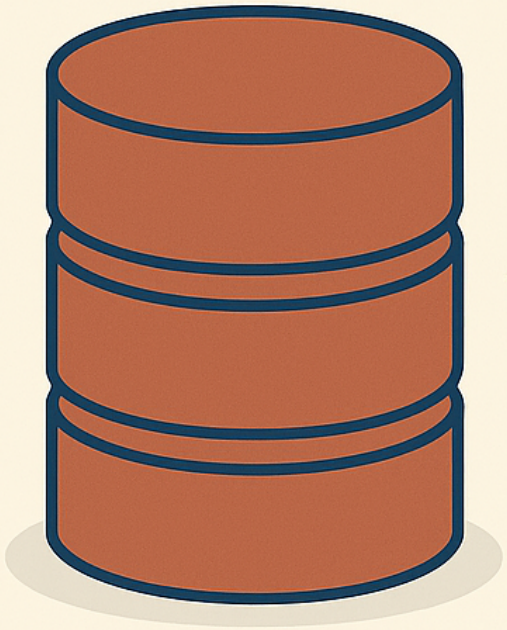

Modern Oracle Database Programming
Level Up Your Skill Set to Oracle's Latest and Most Powerful Features in SQL, PL/SQL, and JSON
—
Alex Nuijten
Patrick Barel
Foreword by Chris Saxon
Apress®

HR DATA MODEL



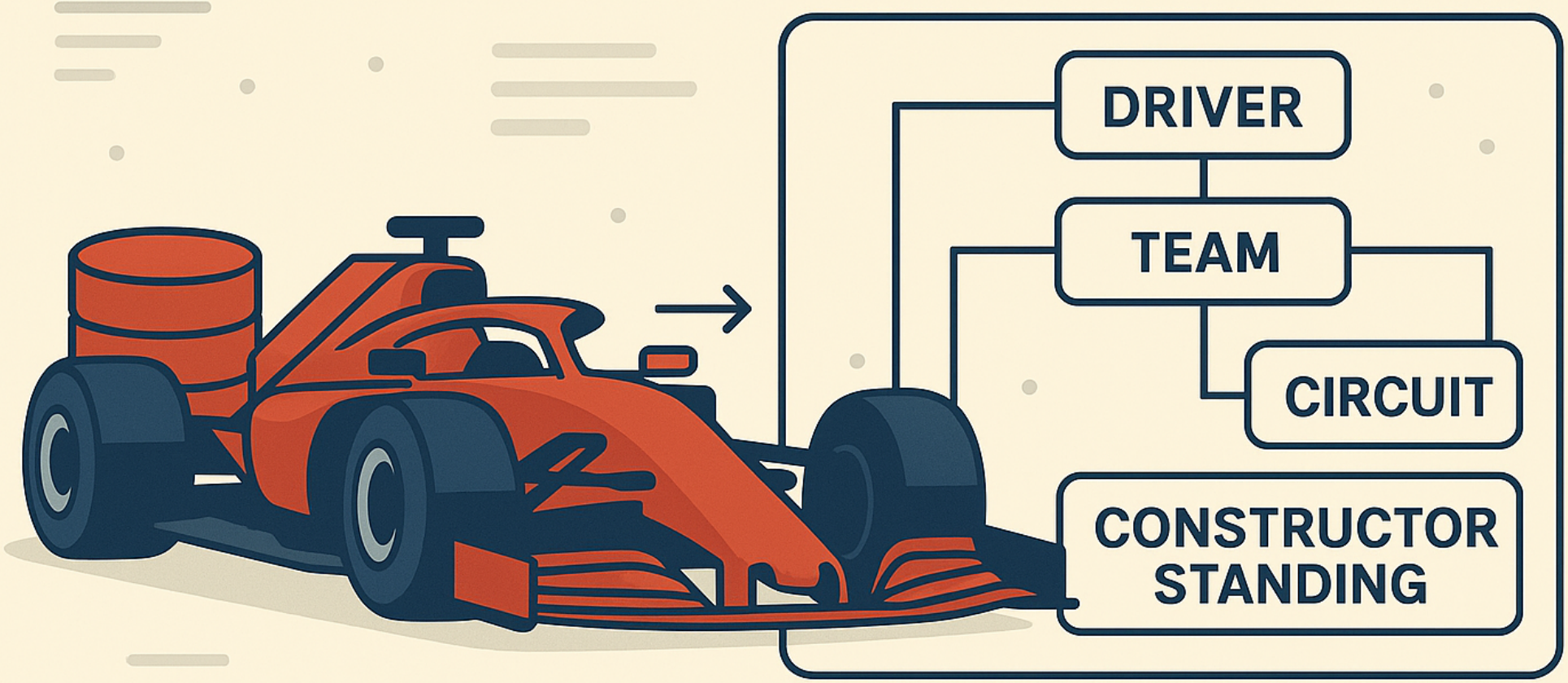
HR DATA MODEL

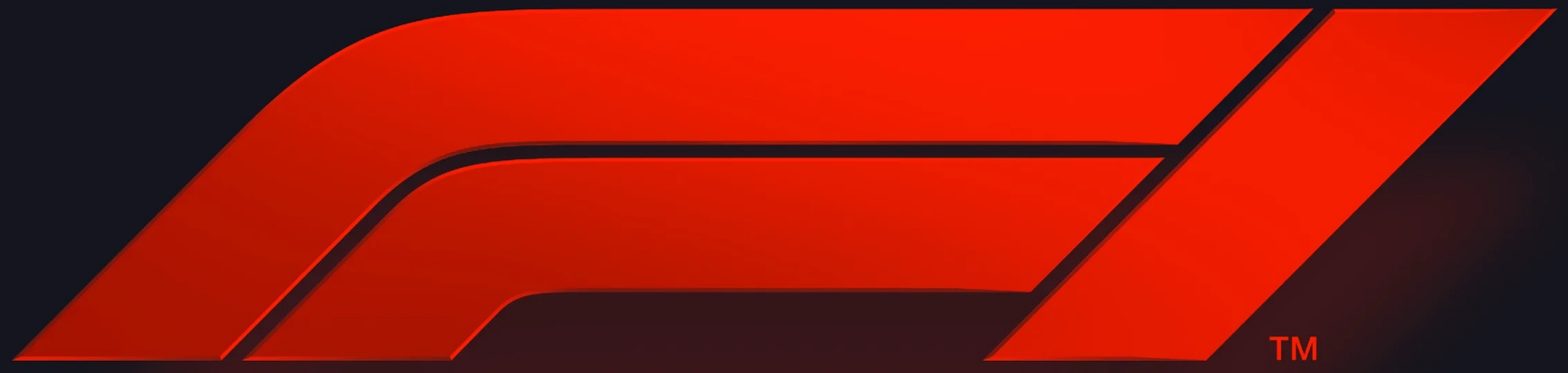
ORDERS DATA MODEL

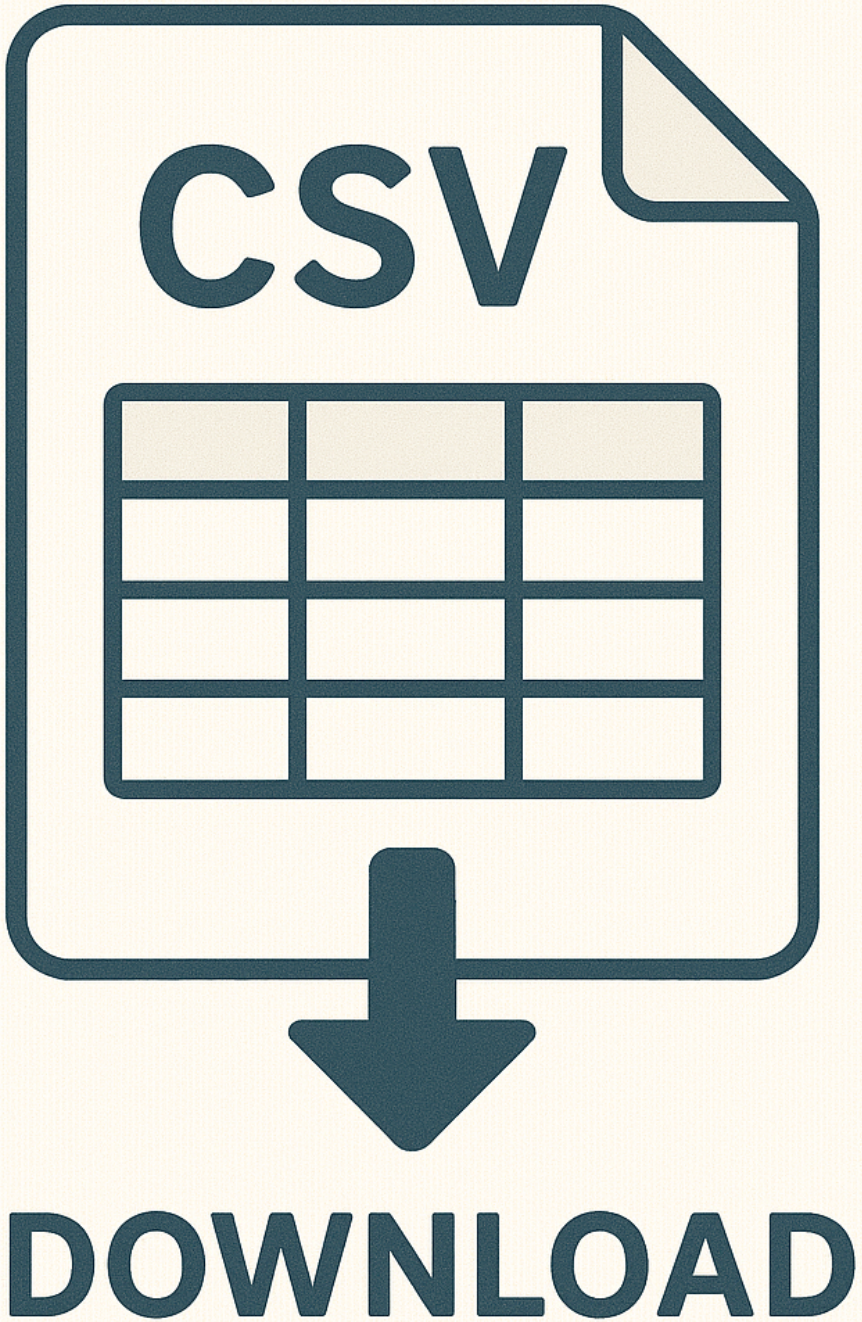


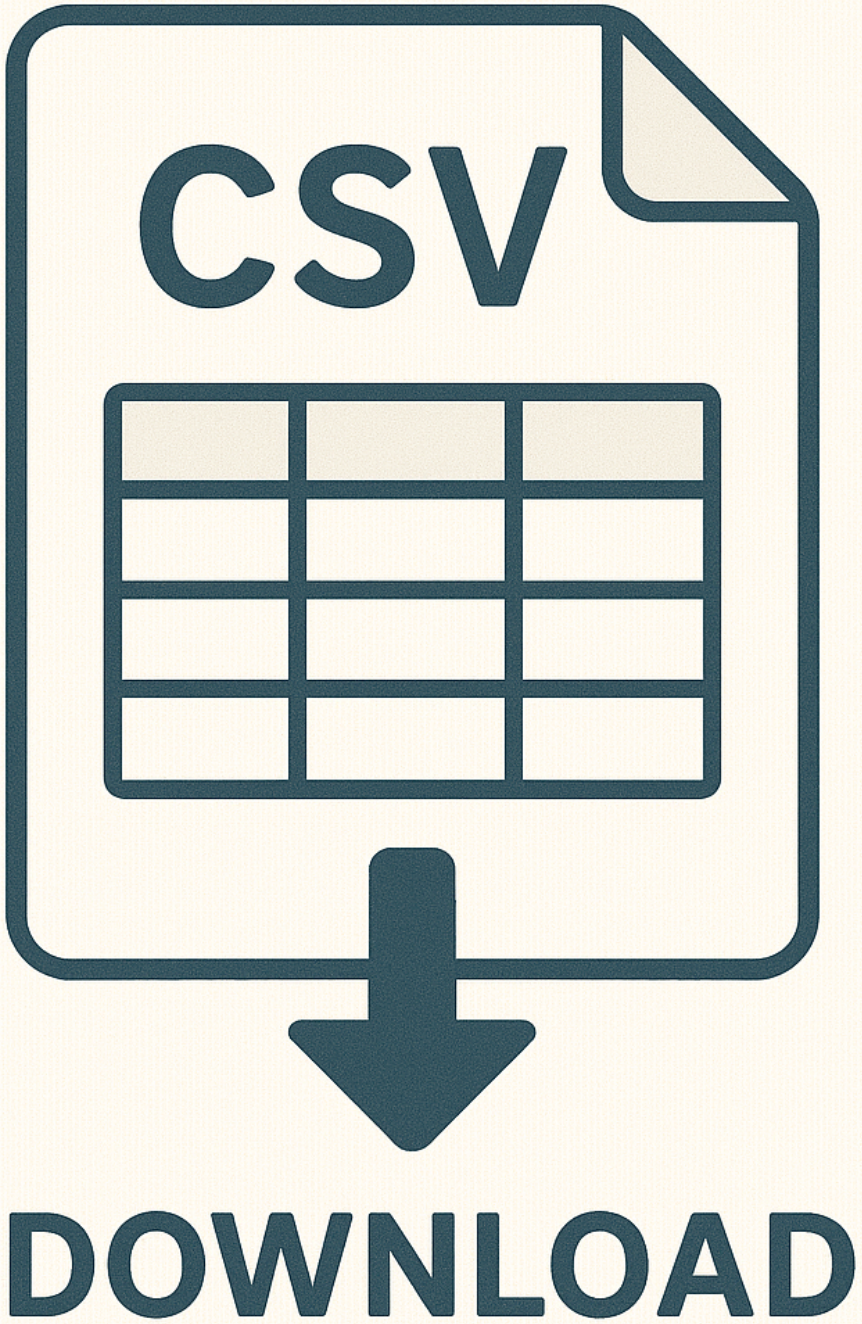
Order_ID	
Order_2	Amount
1	99
2	150
3	75

FORMULA 1
DATA MODEL

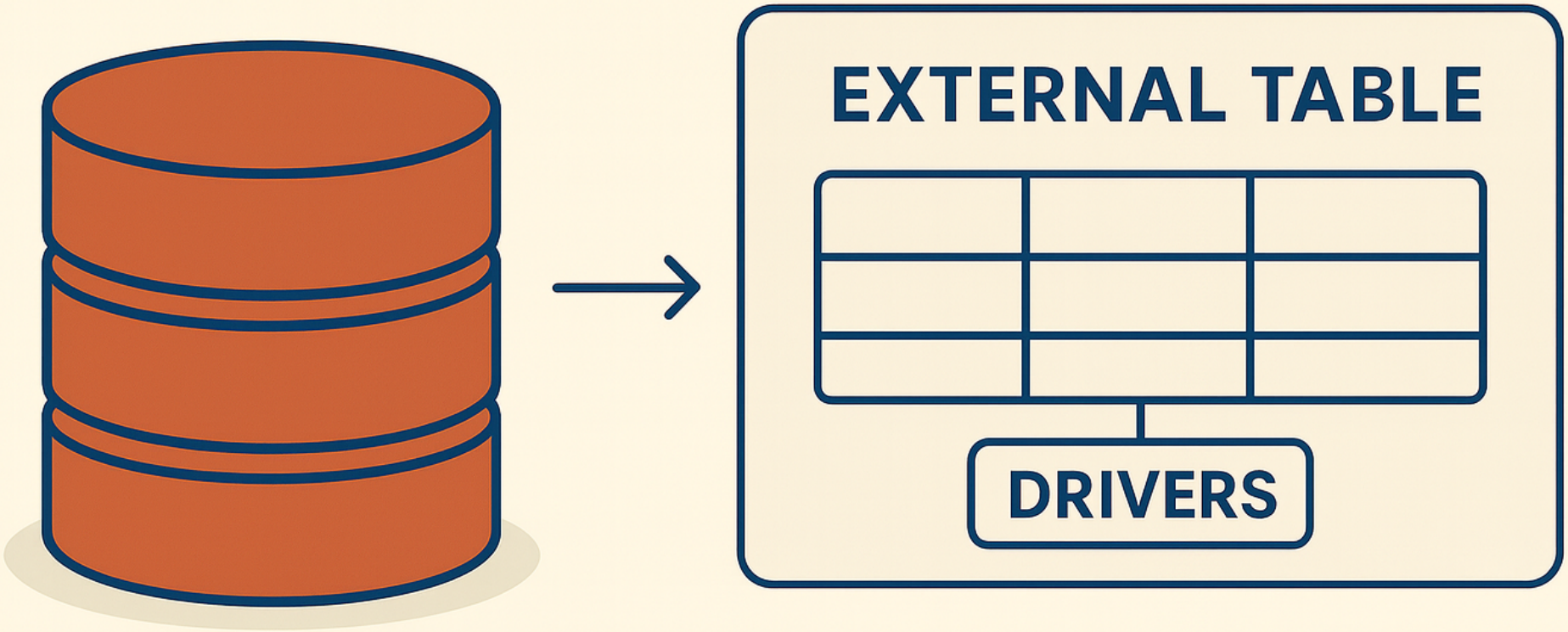



TM


CSV
DOWNLOAD


CSV
DOWNLOAD

ORACLE 19c DATABASE



EXTERNAL TABLE

```
create table drivers_ext ( driverid      number   (  11      )
                         , driverref     varchar2( 256 char )
                         , driver_number varchar2( 256 char )
                         , code          varchar2(   3 char )
                         , forename      varchar2( 256 char )
                         , surname       varchar2( 256 char )
                         , dob           date
                         , nationality   varchar2( 256 char )
                         , url           varchar2( 256 char )
                         ) organization external
                         ( type oracle_loader
                           default directory external_tables
                           access parameters ( records delimited by newline
                                               skip 1
                                               badfile 'drivers.bad'
                                               logfile 'drivers.log'
                                               fields terminated by ','
                                               optionally enclosed by '"'
                                               ( driverid
                                               , driverref
                                               , driver_number
                                               , code
                                               , forename
                                               , surname
                                               , dob char( 10 ) date_format date mask "YYYY-MM-DD"
                                               , nationality
                                               , url
                                               )
                                             )
                           location ( 'drivers.csv' )
                         )
reject limit 0
```

DRIVERS

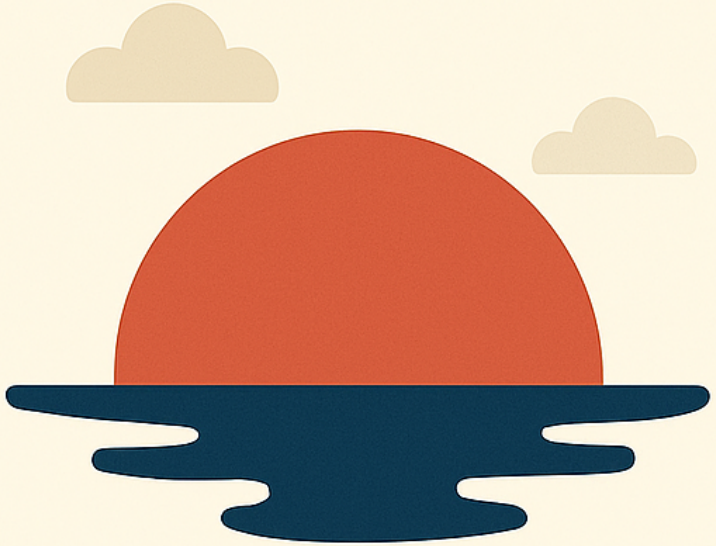
driverid	max_verstappen
name	Verstappen
dob	30-sep-1997
nationality	Dutch

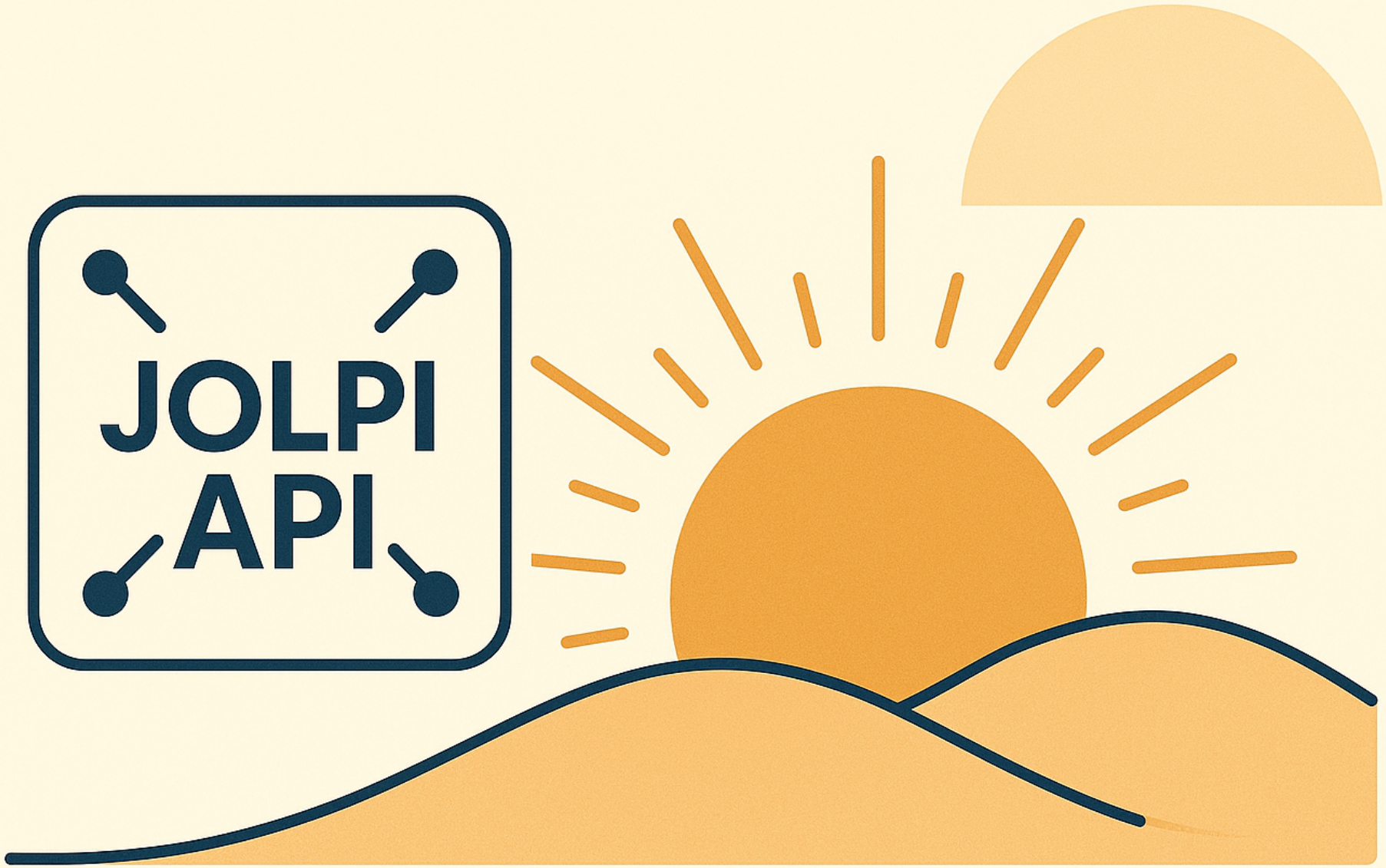
```
create table drivers (
  driverid                number  (   11      ) generated by default on null as identity
, driverref               varchar2(  255 char )
, driver_number           number  (   11      )
, code                    varchar2(    3 char )
, forename                varchar2(  255 char )
, surname                 varchar2(  255 char )
, dob                     date
, nationality             varchar2(  255 char )
, url                     varchar2(  255 char )
, constraint pk_drivers   primary key (driverid)
)
```

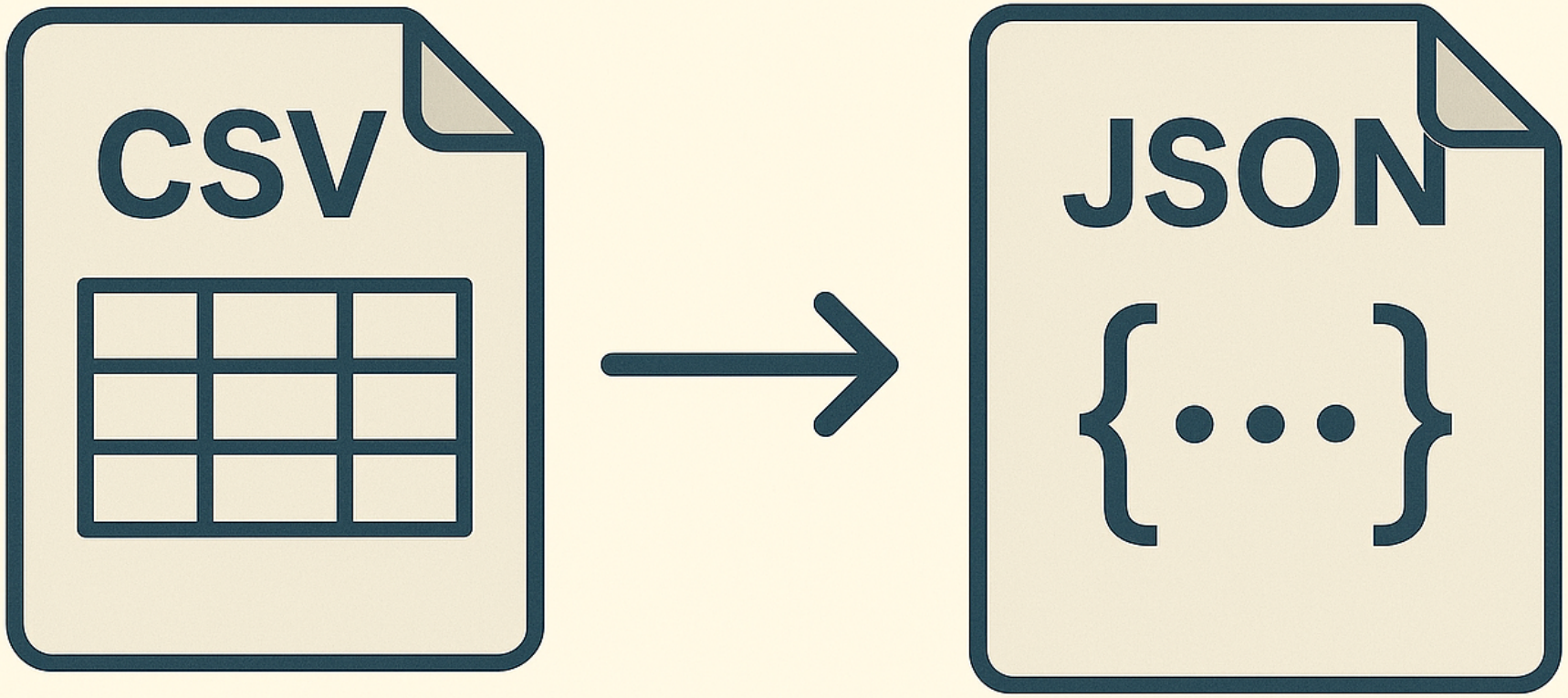
```
insert into drivers
select drex.driverid
     , drex.driverref
     , case
         when drex.driver_number = '\N' then null
         else drex.driver_number
       end driver_number
     , case
         when drex.code = '\N' then null
         else drex.code
       end code
     , drex.forename
     , drex.surname
     , drex.dob
     , drex.nationality
     , drex.url
from   drivers_ext drex
```

```
855 rows inserted.
```


ERGAST
F1
DATASET




JOLPI
API


CSV
JSON
{...}

```
{
  "MRData": {
    "xmlns": "",
    "series": "f1",
    "url": "https://api.jolpi.ca/ergast/f1/2025/drivers/",
    "limit": "30",
    "offset": "0",
    "total": "21",
    "DriverTable": {
      "season": "2025",
      "Drivers": [
        {
          "driverId": "max_verstappen",
          "permanentNumber": "33",
          "code": "VER",
          "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
          "givenName": "Max",
          "familyName": "Verstappen",
          "dateOfBirth": "1997-09-30",
          "nationality": "Dutch"
        },
        {
          "driverId": "alonso",
          "permanentNumber": "14",
          "code": "ALO",
          "url": "http://en.wikipedia.org/wiki/Fernando_Alonso",
          "givenName": "Fernando",
          "familyName": "Alonso",
          "dateOfBirth": "1981-07-29",
          "nationality": "Spanish"
        },
        {
          "driverId": "antonelli",
          "permanentNumber": "12",
```

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	**max_verstappen**	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```

DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	**Verstappen**	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```

DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```

DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "driverId": "max_verstappen",
  "permanentNumber": "33",
  "code": "VER",
  "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
  "givenName": "Max",
  "familyName": "Verstappen",
  "dateOfBirth": "1997-09-30",
  "nationality": "Dutch"
}
```


DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
select json_object
{
  "driverId": "max_verstappen",driver.driverref
, "permanentNumber": "33",driver_number
, "code": "VER",    : driv.code
, "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
, "givenName": "Max",driv.forename
, "familyName": "Verstappen",name
, "dateOfBirth": "1997-09-30",
, "nationality": "Dutch"v.nationality
}
from  drivers driv
```

DRIVERID	DRIVERREF	DRIVER_NUMBER	CODE	FORENAME	SURNAME	DOB	NATIONALITY	URL
830	max_verstappen	33	VER	Max	Verstappen	1997-09-30	Dutch	http://en.wikipedia.org/wiki/Max_Verstappen

```
{
  "MRData": {
    "xmlns": "",
    "series": "f1",
    "url": "https://api.jolpi.ca/ergast/f1/2025/drivers/",
    "limit": "30",
    "offset": "0",
    "total": "21",
    "DriverTable": {
      "season": "2025",
      "Drivers": [
        {
          "driverId": "max_verstappen",
          "permanentNumber": "33",
          "code": "VER",
          "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
          "givenName": "Max",
          "familyName": "Verstappen",
          "dateOfBirth": "1997-09-30",
          "nationality": "Dutch"
        },
        {
          "driverId": "alonso",
          "permanentNumber": "14",
          "code": "ALO",
          "url": "http://en.wikipedia.org/wiki/Fernando_Alonso",
          "givenName": "Fernando",
          "familyName": "Alonso",
          "dateOfBirth": "1981-07-29",
          "nationality": "Spanish"
        },
        {
          "driverId": "antonelli",
          "permanentNumber": "12",
```

```
{
  "MRData": {
    "xmlns": "",
    "series": "f1",
    "url": "https://api.jolpi.ca/ergast/f1/2025/drivers/",
    "limit": "30",
    "offset": "0",
    "total": "21",
    "DriverTable": {
      "season": "2025",
      "Drivers": [
        {
          "driverId": "max_verstappen",
          "permanentNumber": "33",
          "code": "VER",
          "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
          "givenName": "Max",
          "familyName": "Verstappen",
          "dateOfBirth": "1997-09-30",
          "nationality": "Dutch"
        },
        {
          "driverId": "alonso",
          "permanentNumber": "14",
          "code": "ALO",
          "url": "http://en.wikipedia.org/wiki/Fernando_Alonso",
          "givenName": "Fernando",
          "familyName": "Alonso",
          "dateOfBirth": "1981-07-29",
          "nationality": "Spanish"
        },
        {
          "driverId": "antonelli",
          "permanentNumber": "12",
```

```
{
  MRData : {
    "xmlns": "",
    "series": "f1",
    "url": https://api.jolpi.ca/ergast/f1/2025/drivers/,
    "limit": "30",
    "offset": "0",
    "total": "21",
    DriverTable : {
      "season": "2025",
      Drivers : [
        {
          "driverId": "max_verstappen",
          "permanentNumber": "33",
          "code": "VER",
          "url": "http://en.wikipedia.org/wiki/Max_Verstappen",
          "givenName": "Max",
          "familyName": "Verstappen",
          "dateOfBirth": "1997-09-30",
          "nationality": "Dutch"
        },
        {
          "driverId": "alonso",
          "permanentNumber": "14",
          "code": "ALO",
          "url": "http://en.wikipedia.org/wiki/Fernando_Alonso",
          "givenName": "Fernando",
          "familyName": "Alonso",
          "dateOfBirth": "1981-07-29",
          "nationality": "Spanish"
        },
        {
          "driverId": "antonelli",
          "permanentNumber": "12",
```

```
drop table if exists driverdocs purge


create table if not exists driverdocs
( data json )



insert into driverdocs
select jsta.driverdoc
from   ( select apex_web_service.make_rest_request_b(
           p_url => 'https://api.jolpi.ca/ergast/f1/2025/drivers/?format=json'
         , p_http_method => 'GET'
         ) as doc
       ) webs
cross
apply  json_table ( webs.doc, '$.MRData.DriverTable.Drivers[*]'
                        columns ( driverdoc clob format json path '$' )
                 ) jsta
```

```
drop table if exists driverdocs purge


create table if not exists driverdocs
( data json )



insert into driverdocs
select jsta.driverdoc
from   ( select apex_web_service.make_rest_request_b(
           p_url => 'https://api.jolpi.ca/ergast/f1/2025/drivers/?format=json'
         , p_http_method => 'GET'
         ) as doc
       ) webs
cross
apply  json_table ( webs.doc, '$.MRData.DriverTable.Drivers[*]'
                        columns ( driverdoc clob format json path '$' )
                 ) jsta
```

```
alter table drivers modify driverid number (   11 ) generated by default on null as identity
                                                   ( start with 1000 )
```

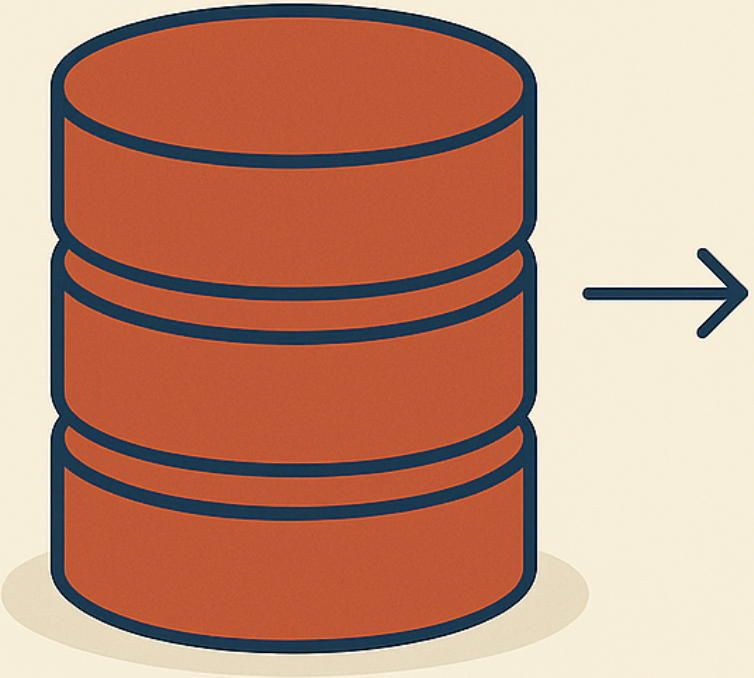
```
alter table drivers drop constraint pk_drivers
```

```
alter table drivers add constraint pk_drivers primary key ( driverref )
```

```
begin
  dbms_errlog.create_error_log( dml_table_name => 'drivers' );
end;
/
```

```
insert into drivers(
  driverref
, driver_number
, code
, forename
, surname
, dob
, nationality
, url
)
select driverId driverref
     , permanentNumber driver_number
     , code
     , givenName forename
     , familyName surname
     , dateOfBirth dob
     , nationality
     , url
from   driverdocs drdo
cross  apply
       json_table( drdo.data
                 , '$' columns ( driverId        varchar2(256) path '$.driverId'
                               , permanentNumber varchar2(256) path '$.permanentNumber'
                               , code            varchar2(256) path '$.code'
                               , givenName       varchar2(256) path '$.givenName'
                               , familyName      varchar2(256) path '$.familyName'
                               , dateOfBirth     timestamp     path '$.dateOfBirth'
                               , nationality     varchar2(256) path '$.nationality'
                               , url             varchar2(256) path '$.url'
                               )
                 )
where  json_exists( drdo.data, '$.driverId' )
log    errors into err$_drivers( 'INSERT' ) reject limit unlimited
```

CONSTRUCTORS



constructorid	red_bull
name	Red Bull
nationality	Austrian

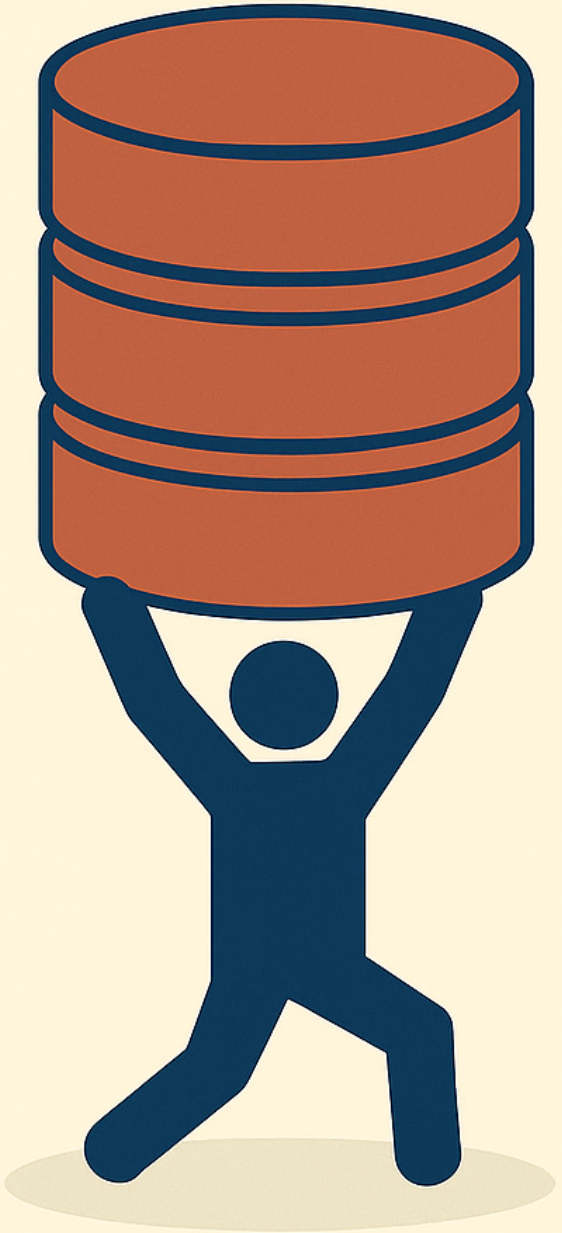
```
select apex_web_service.make_rest_request_b
       ( p_url => 'https://api.jolpi.ca/ergast/f1/2025/constructors/?format=json'
       , p_http_method => 'GET'
       ) doc
from   dual
```

```
with webservicedocs as
( select apex_web_service.make_rest_request_b
         ( p_url => 'https://api.jolpi.ca/ergast/f1/2025/constructors/?format=json'
         , p_http_method => 'GET'
         ) doc
  from   dual
)
select jsta.constructordoc
from   webservicedocs webs
cross
apply  json_table ( webs.doc, '$.MRData.ConstructorTable.Constructors[*]'
                            columns ( constructordoc clob format json path '$' )
                  ) jsta
```

```
with webservicedocs as
( select apex_web_service.make_rest_request_b
         ( p_url => 'https://api.jolpi.ca/ergast/f1/2025/constructors/?format=json'
         , p_http_method => 'GET'
         ) doc
  from   dual
)
select jsta.constructordoc
from   webservicedocs webs
cross
apply  json_table ( webs.doc, '$.MRData.ConstructorTable.Constructors[*]'
                            columns ( constructordoc clob format json path '$' )
                  ) jsta
```

```
with webservicedocs as
( select apex_web_service.make_rest_request_b
         ( p_url => 'https://api.jolpi.ca/ergast/f1/2025/constructors/?format=json'
         , p_http_method => 'GET'
         ) doc
  from   dual
)
select jsta.constructordoc
from   webservicedocs webs
cross
apply  json_table ( webs.doc, '$.MRData.ConstructorTable.Constructors[*]'
                              columns ( constructordoc clob format json path '$' )
                  ) jsta
```

```
create table if not exists constructors
( constructorid varchar2( 64 char )
, name          varchar2( 64 char )
, nationality   varchar2( 64 char )
, url           varchar2( 64 char )
, constraint pk_constructors primary key ( constructorid )
)
```



```
{_id: 1
 Name: 'Red Bull'
```

iD	Name
1	Red Bull

ORACLE 23ai

JSON RELATIONAL DUALITY VIEWS

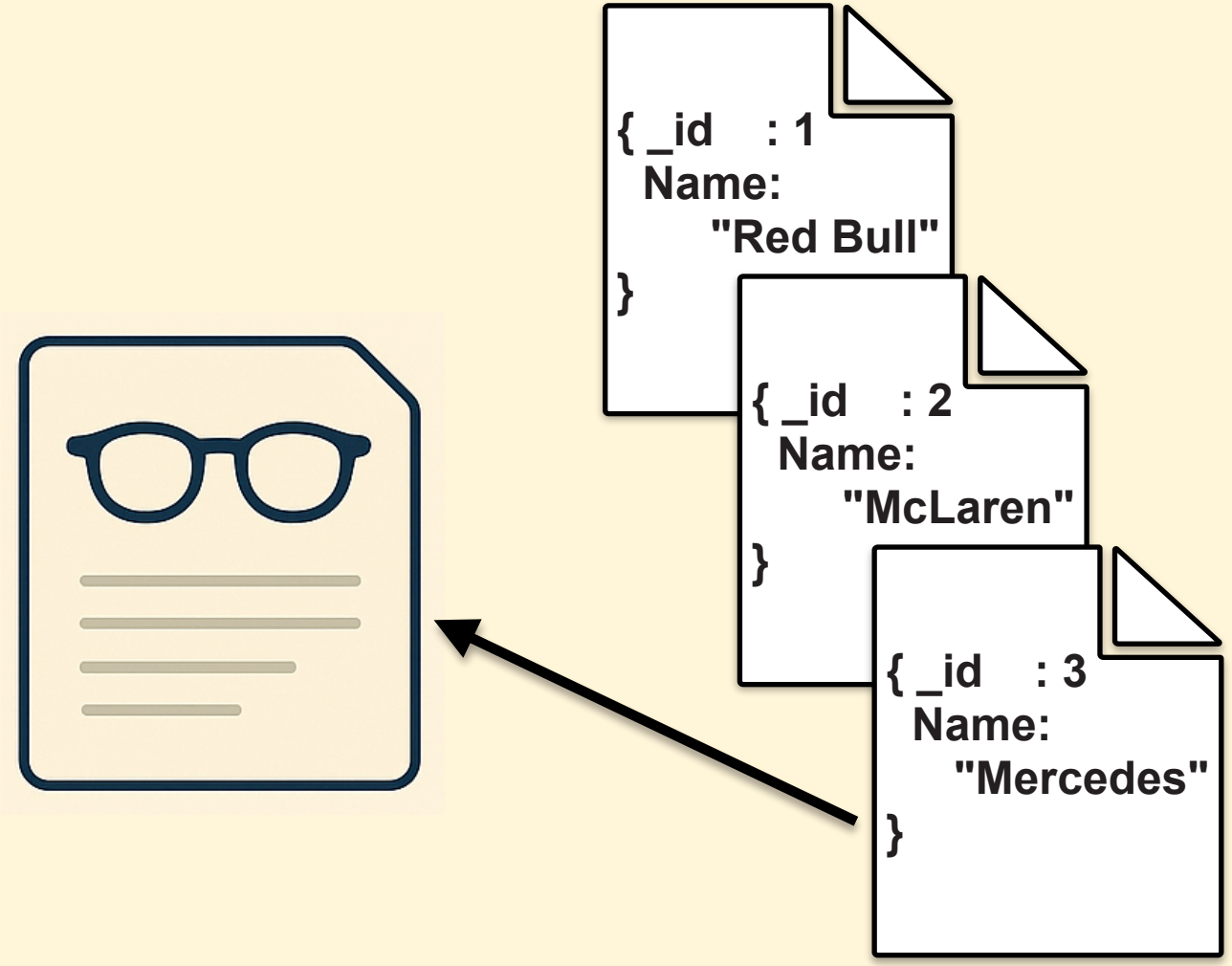
Id	Name
1	Red Bull
2	McLaren

Id	Name
1	Red Bull
2	McLaren

```
{ _id    : 1
  Name:
        "Red Bull"
}
```

```
{ _id    : 2
  Name:
        "McLaren"
}
```

Id	Name
1	Red Bull
2	McLaren



Id	Name
1	Red Bull
2	McLaren

```
{ _id    : 1
  Name:
       "Red Bull"
}
```

```
{ _id    : 2
  Name:
       "McLaren"
}
```

```
{ _id    : 3
  Name:
     "Mercedes"
}
```

Id	Name
1	Red Bull
2	McLaren
3	Mercedes

```
{ _id    : 1
  Name:
       "Red Bull"
}
```

```
{ _id    : 2
  Name:
       "McLaren"
}
```

```
{ _id    : 3
  Name:
     "Mercedes"
}
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  'constructorId' : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
ORA-42647: Missing '_id' field at the root level for JSON-relational duality view 'CONSTRUCTORS_DV'.
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
View CONSTRUCTORS_DV created.
```

```
insert into constructors_dv
values ('{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}')
```

```
ORA-40944: Cannot insert into JSON Relational Duality View 'CONSTRUCTORS_DV':
           The input JSON document is invalid.
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
{
  '_id'            : c.constructorid
, 'url'            : c.url
, 'name'           : c.name
, 'nationality'    : c.nationality
}
```

```
{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}
```

```
{
  '_id'            : c.constructorid
, 'url'            : c.url
, 'name'           : c.name
, 'nationality'    : c.nationality
}
```

```
{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}
```

```
{                                                    {
  '_id'            : c.constructorid   ✗   "constructorId": "red_bull",
, 'url'            : c.url             ✓   "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
, 'name'           : c.name            ✓   "name": "Red Bull",
, 'nationality'    : c.nationality     ✓   "nationality": "Austrian"
}                                                    }
```

```
{
  '_id'            : c.constructorid
, 'url'            : c.url
, 'name'           : c.name
, 'nationality'    : c.nationality
}
```

```
{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}
```

```
{
  '_id'            : c.constructorid
, 'url'            : c.url
, 'name'           : c.name
, 'nationality'    : c.nationality
}
```

```
{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'constructorId' : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
ORA-44971: JSON relational duality view cannot have duplicate column 'CONSTRUCTORS'.'CONSTRUCTORID' specified.
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'constructorId' : c.constructorid
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
ORA-44971: JSON relational duality view cannot have duplicate column 'CONSTRUCTORS'.'CONSTRUCTORID' specified.

https://docs.oracle.com/error-help/db/ora-44971/
44971. 00000 -  "JSON relational duality view cannot have duplicate column '%s'.'%s' specified."
*Cause:    Duplicate columns were specified for a JSON relational duality view.
*Action:   Remove duplicate columns.
```

```
create or replace json relational duality view constructors_dv
as
  select json
{
  '_id'           : c.constructorid
, 'constructorId' : generated using ( c.constructorid )
, 'url'           : c.url
, 'name'          : c.name
, 'nationality'   : c.nationality
}
  from   constructors c with insert update delete
```

```
View CONSTRUCTORS_DV created.
```

```
insert into constructors_dv
values ('{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}')
```

```
ORA-42692: Cannot insert into JSON Relational Duality View 'HOWTO'.'CONSTRUCTORS_DV':
           Error while inserting into table 'CONSTRUCTORS'

ORA-01400: cannot insert NULL into ("HOWTO"."CONSTRUCTORS"."CONSTRUCTORID")
```

```
create or replace trigger constructors_dv_bri
before insert
on constructors_dv
for each row
declare
  l_jsonobj json_object_t;
begin
  -- get the new JSON document into a local variable
  l_jsonobj := json_object_t( :new.data );
  -- check if the key '_id' is present
  if not l_jsonobj.has( '_id' ) then
    -- if not, then put the value of the constructorId into the _id key and add it to the document
    l_jsonobj.put( '_id', l_jsonobj.get( 'constructorId' ) );
    -- replace the new JSON document with the (edited) value
    :new.data := l_jsonobj.to_json;
  end if;
end;
/
```

```
insert into constructors_dv
values ('{
  "constructorId": "red_bull",
  "url": "http://en.wikipedia.org/wiki/Red_Bull_Racing",
  "name": "Red Bull",
  "nationality": "Austrian"
}')
```

```
1 row inserted.
```

```
insert
into   constructors_dv
select data
from   constructordocs
```

```
10 rows inserted.
```

WHICH IS BETTER?

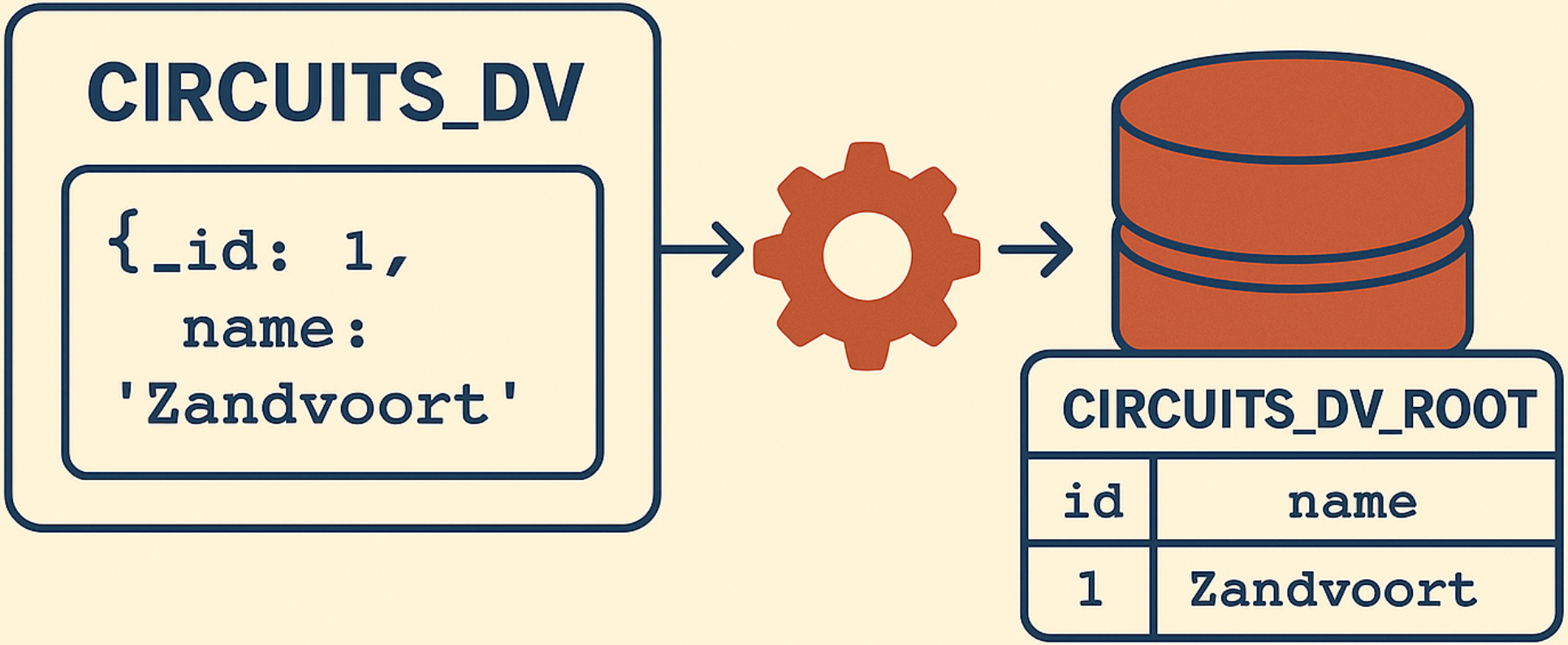
VERBOSE SQL STATEMENT

```
insert into constructors
(
  constructorid
, name
, nationality
, url
)
select constructorId
     , name
     , nationality
     , url
from   constructordocs codo
cross  apply
       json_table( codo.data, '$' columns (
  constructorId varchar2(256) path '$.constructorId'
, name          varchar2(256) path '$.name'
, nationality   varchar2(256) path '$.nationality'
, url           varchar2(256) path '$.url' )
                 )
where json_exists( codo.data, '$.constructorId' )
```

JSON RELATIONAL DUALITY VIEW

```
insert
into   constructors_dv
select data
from   constructordocs
```

With some code in a trigger


CIRCUITS_DV
{_id: 1,
name:
'Zandvoort'
CIRCUITS_DV_ROOT
id
name
1
Zandvoort

```
drop table if exists circuitdocs purge


create table if not exists circuitdocs
( data json )



insert into circuitdocs
select jsta.circuitdoc
from   ( select apex_web_service.make_rest_request_b(
           p_url => 'https://api.jolpi.ca/ergast/f1/2025/circuits/?format=json'
         , p_http_method => 'GET'
         ) as doc
       ) webs
cross
apply  json_table ( webs.doc, '$.MRData.CircuitTable.Circuits[*]'
                        columns ( circuitdoc clob format json path '$' )
                 ) jsta
```

```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"]
           }'
        )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```

```
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"]
           }'
          )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```

```
{ "outputFormat"   : "executable"
, "sourceSchema"   : "HOWTO"
, "tableNames"     : [ "CIRCUITDOCS" ]
, "targetSchema"   : "HOWTO"
, "updatability"   : true
, "useFlexFields"  : false
, "viewNames"      : [ "CIRCUITS_DV"]
}
```

```
{ "outputFormat"   : "executable"
, "sourceSchema"   : "HOWTO"
, "tableNames"     : [ "CIRCUITDOCS" ]
, "targetSchema"   : "HOWTO"
, "updatability"   : true
, "useFlexFields"  : false
, "viewNames"      : [ "CIRCUITS_DV"]
}
```

Field	Value
tableNames	JSON array of strings of the table names of JSON collection to be processed
viewNames	JSON array of strings of the output duality view names to be generated
useFlexFields	Boolean field which indicates whether flex fields should be used in the duality views. • When `true`, for each duality view, a flex column is added to each table that directly underlies the top-level fields of an object in the supported documents. • Default value for this field is `true`.
updatability	Boolean field which indicates whether the duality views created by the migrator should be updatable by default. • When `true`, the migrator will mark all duality views as updatable (i.e., annotations will be set for maximum updatability of the duality view). • When `false`, all duality views created will be read-only. • Default value for this field is `true`.
normalize	Boolean field which indicates whether the migrator will attempt to normalize the relational tables it infers for de-duplicating data. The default value for this parameter is 'true'

```
{ "outputFormat"   : "executable"
, "sourceSchema"   : "HOWTO"
, "tableNames"     : [ "CIRCUITDOCS" ]
, "targetSchema"   : "HOWTO"
, "updatability"   : true
, "useFlexFields"  : false
, "viewNames"      : [ "CIRCUITS_DV"]
}
```

Field	Value
sourceSchema	This string field defines the schema name for the input tables. • Default value for this field is the current user schema.
targetSchema	This string field defines the schema name for the output tables and duality views. • There is no default value for this field. • If not provided then no database schema is specified in the output DDL; the names of the database objects to be created are unqualified. This means that the schema used is the one that's current at the time the DDL code is executed (not when it is generated).
tablespace	This string field defines the `tablespace` name to use for all tables created by the migrator. • This field has no default. • If this field is not specified, then the `tablespace` for tables created by the migrator will remain unspecified.
ingestLimit	This numeric field defines the maximum number of documents that will be analyzed in each JSON collection to infer the relational schema. • The default value of this parameter is 100000.
outputFormat	This string field defines the format of the output DDL script. • The possible values for this field are "standalone" and "executable".

```
{ "outputFormat"   : "executable"
, "sourceSchema"   : "HOWTO"
, "tableNames"     : [ "CIRCUITDOCS" ]
, "targetSchema"   : "HOWTO"
, "updatability"   : true
, "useFlexFields"  : false
, "viewNames"      : [ "CIRCUITS_DV"]
}
```

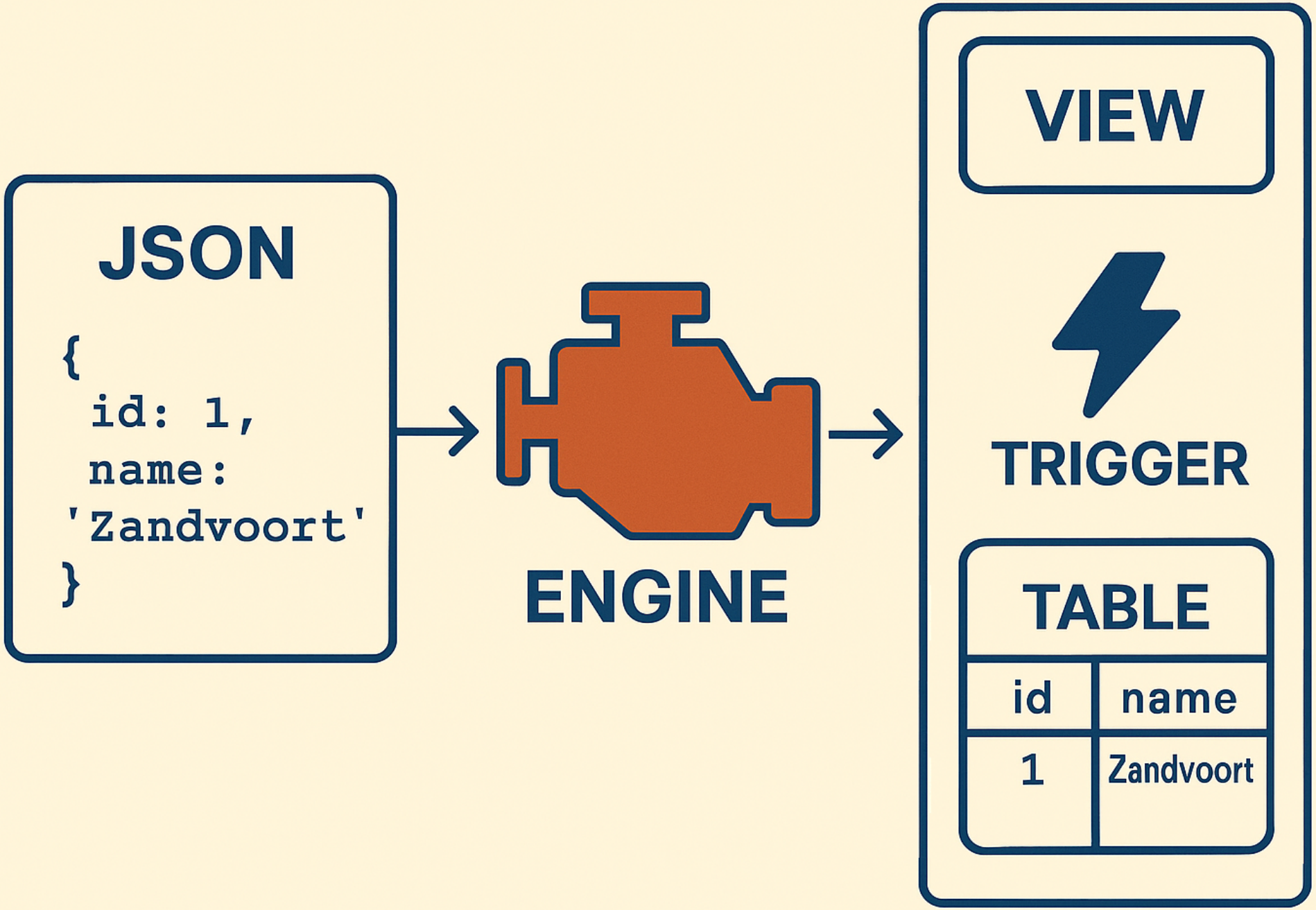
Field	Value
outputFormat	This string field defines the format of the output DDL script. • The possible values for this field are "standalone" and "executable". • The "executable" format creates a DDL script output which can be directly executed using the "execute immediate" PL/SQL construct. • The "standalone" format creates a DDL script which needs to be copied into a SQL file and executed independently. It cannot be executed using the "execute immediate" PL/SQL construct. • The default value of this field is "executable". Note For cases where the output DDL statements can be larger than 32k characters, you MUST use the "standalone" option.
minFieldFrequency	This numeric field defines the threshold in percentage for identifying high entropy fields. • The default value for this field is 5, meaning that fields which appear in less than 5% of documents in a collection will be pruned from the schema.
minTypeFrequency	This numeric field defines the threshold in percentage for identifying high entropy types of a given field. • The default value for this field is 5, meaning that types for a field which appear in less than 5% of documents in a collection will be pruned from the schema.

```
{ "outputFormat"   : "executable"
, "sourceSchema"   : "HOWTO"
, "tableNames"     : [ "CIRCUITDOCS" ]
, "targetSchema"   : "HOWTO"
, "updatability"   : true
, "useFlexFields"  : false
, "viewNames"      : [ "CIRCUITS_DV"]
}
```

Field	Value
softnessThreshold	This numeric field defines the threshold in percentage for allowable 'softness' in identifying functional dependencies in the input data. • The default value of this field is 99, meaning 1% of data could have missing or incorrect information. ○ The value of this field can be set based on the 'dirtiness' of input data. The dirtier (containing more errors) the data, the smaller the value of this field should be.
hints	A JSON array with elements that are JSON objects whose fields specify overrides for the behavior of the converter in generating a relational schema. A hint object must have these fields (otherwise, an error is raised): `type`, `table`, `path`, and `value`. All parameters except `value` are string type. • `type` : A hint can be of type `datatype`, `key` or `normalize`, specifying the type of migrator-behavior override. • `value` : Information specific to the particular `type`, providing detail that defines the hint. (This is the only hint field that is not necessarily a string.) • `table` : A string naming an input table whose document-set data is used to define a duality view. • `path` : A SQL/JSON path expression string that targets data in input JSON documents.

```
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"]
           }'
          )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```

```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"]
           }'
        )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```


JSON
{
id: 1,
name:
'Zandvoort'
}
ENGINE
VIEW
TRIGGER
TABLE
id
name
1
Zandvoort

```
BEGIN
EXECUTE IMMEDIATE 'CREATE TABLE "HOWTO".circuits_dv_root(
   lat  varchar2(64)  /* UNIQUE */,
   url  varchar2(128)  /* UNIQUE */,
   ora$long  varchar2(64)  /* UNIQUE */,
   country  varchar2(64),
   locality  varchar2(64)  /* UNIQUE */,
   circuit_id  varchar2(64)  DEFAULT ON NULL SYS_GUID(),
   circuit_name  varchar2(64)  /* UNIQUE */,
   PRIMARY KEY(circuit_id)
)';
```

```
EXECUTE IMMEDIATE 'CREATE OR REPLACE JSON RELATIONAL DUALITY VIEW "HOWTO".CIRCUITS_DV AS
"HOWTO".circuits_dv_root @insert @update @delete
{
  _id : circuit_id
  url
  Location: "HOWTO".circuits_dv_root @nest
  {
    lat
    long: ora$long
    country
    locality
  }
  circuitId @generated (path: "$._id")
  circuitName: circuit_name
}';
```

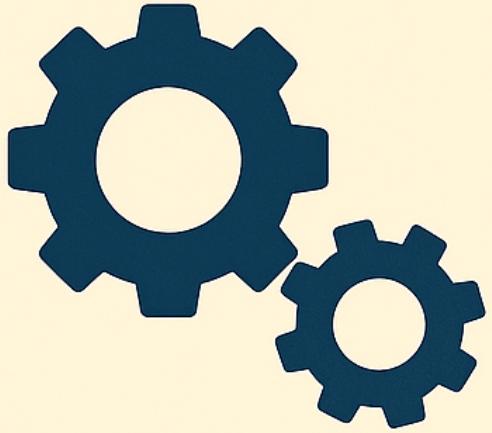
```
EXECUTE IMMEDIATE 'CREATE OR REPLACE TRIGGER "HOWTO".INSERT_TRIGGER_CIRCUITS_DV
  BEFORE INSERT
  ON "HOWTO".CIRCUITS_DV
  FOR EACH ROW
DECLARE
  inp_jsonobj json_object_t;
BEGIN
  inp_jsonobj := json_object_t(:new.data);
  IF NOT inp_jsonobj.has(''_id'')
  THEN
    inp_jsonobj.put(''_id'', inp_jsonobj.get(''circuitId''));
    :new.data := inp_jsonobj.to_json;
  END IF;
END;';
END;
```

AI

~~Artificial Intelligence~~

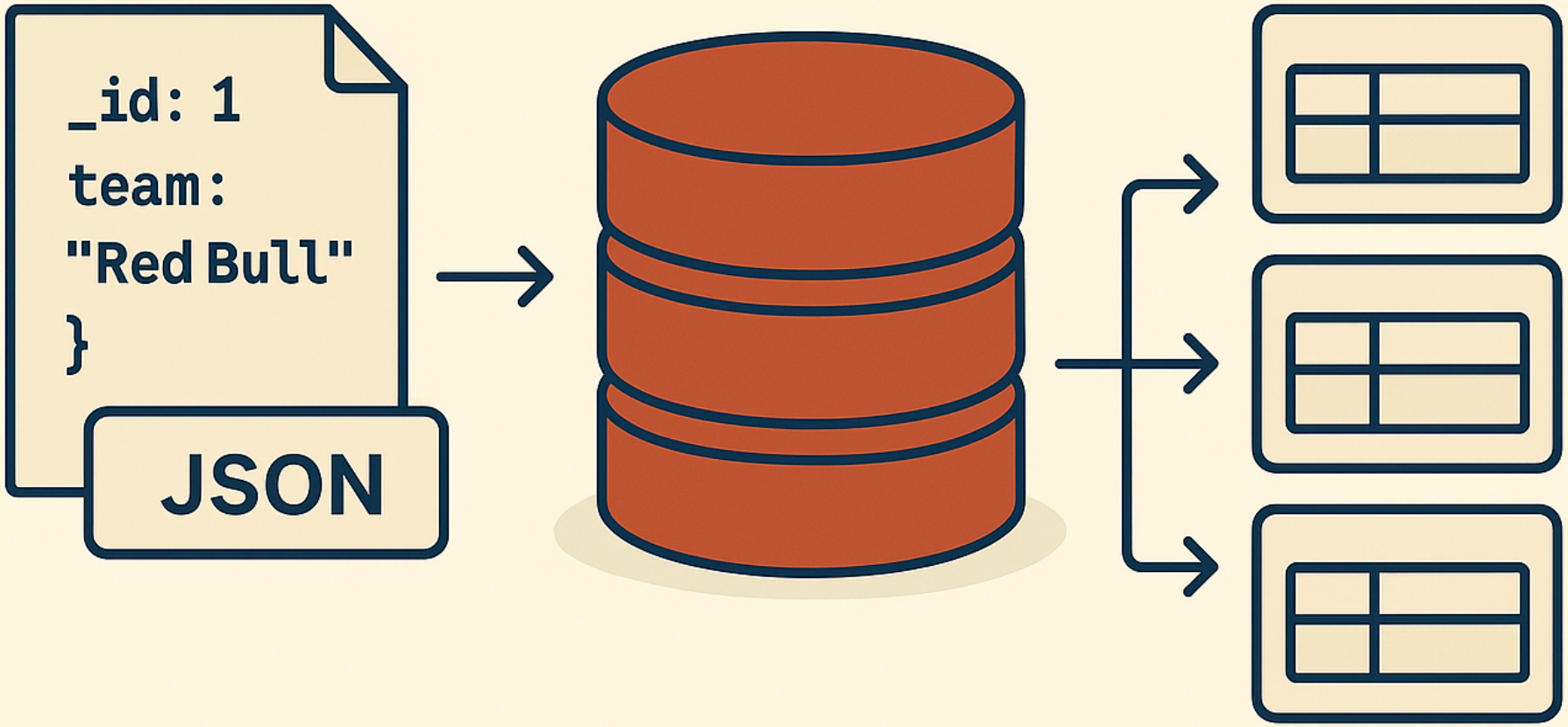
→

Algorithm Implementation

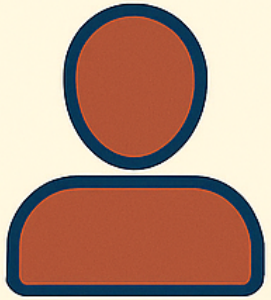


ORACLE 23ai

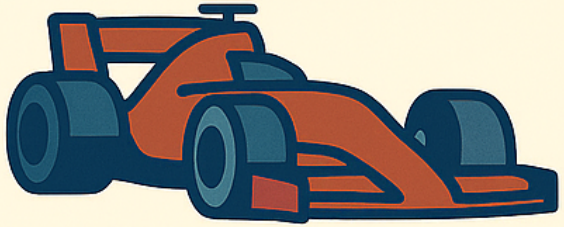
JSON-TO-DUALITY MIGRATOR



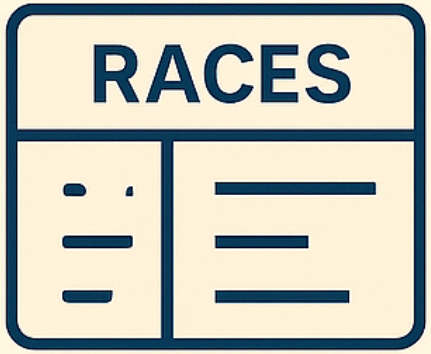
DRIVERS



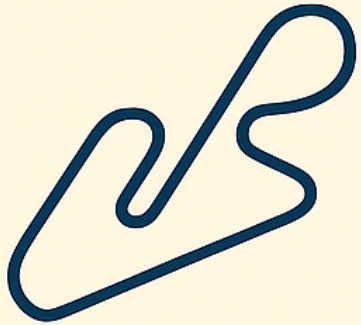
CONSTRUCTORS



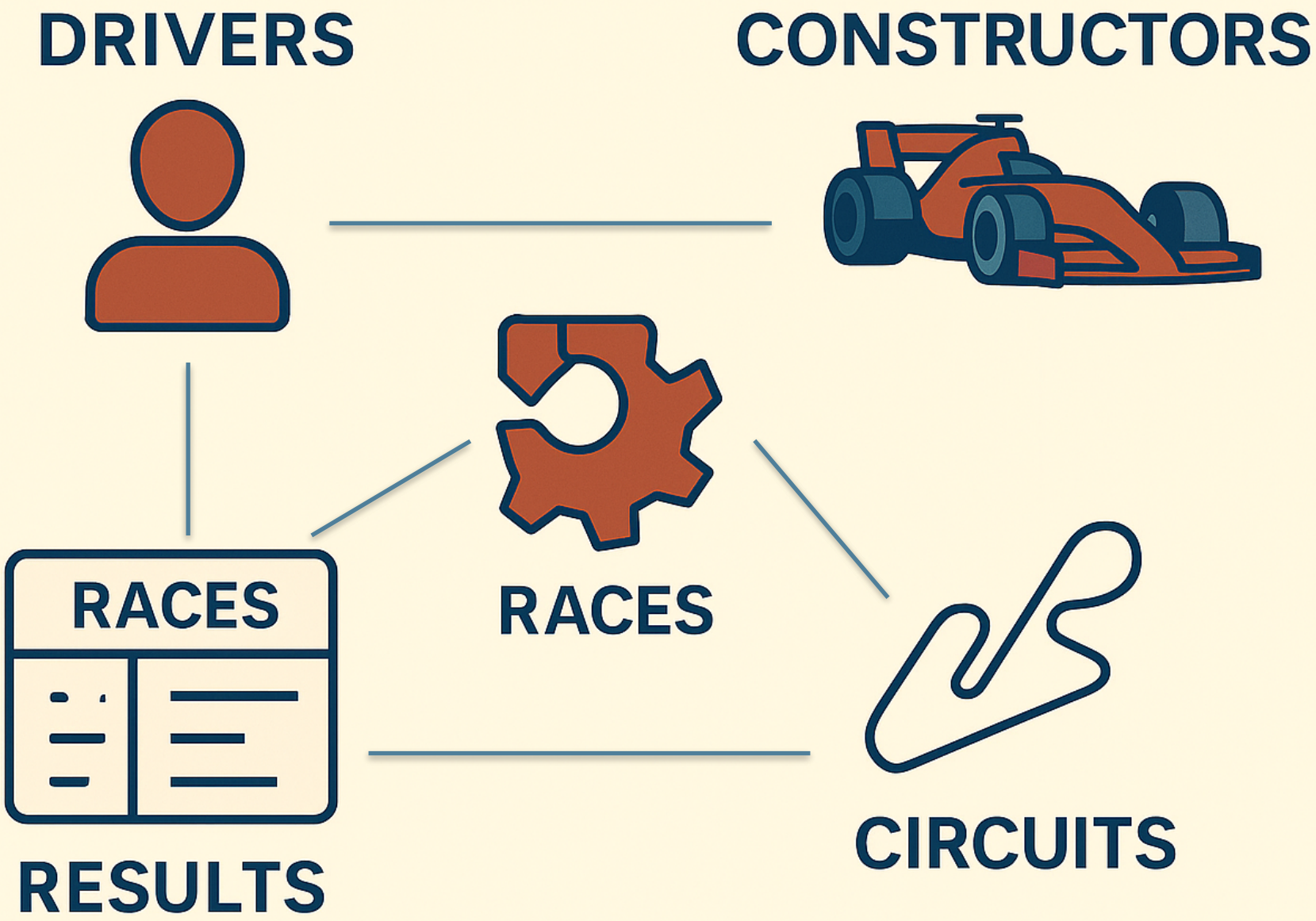
RACES

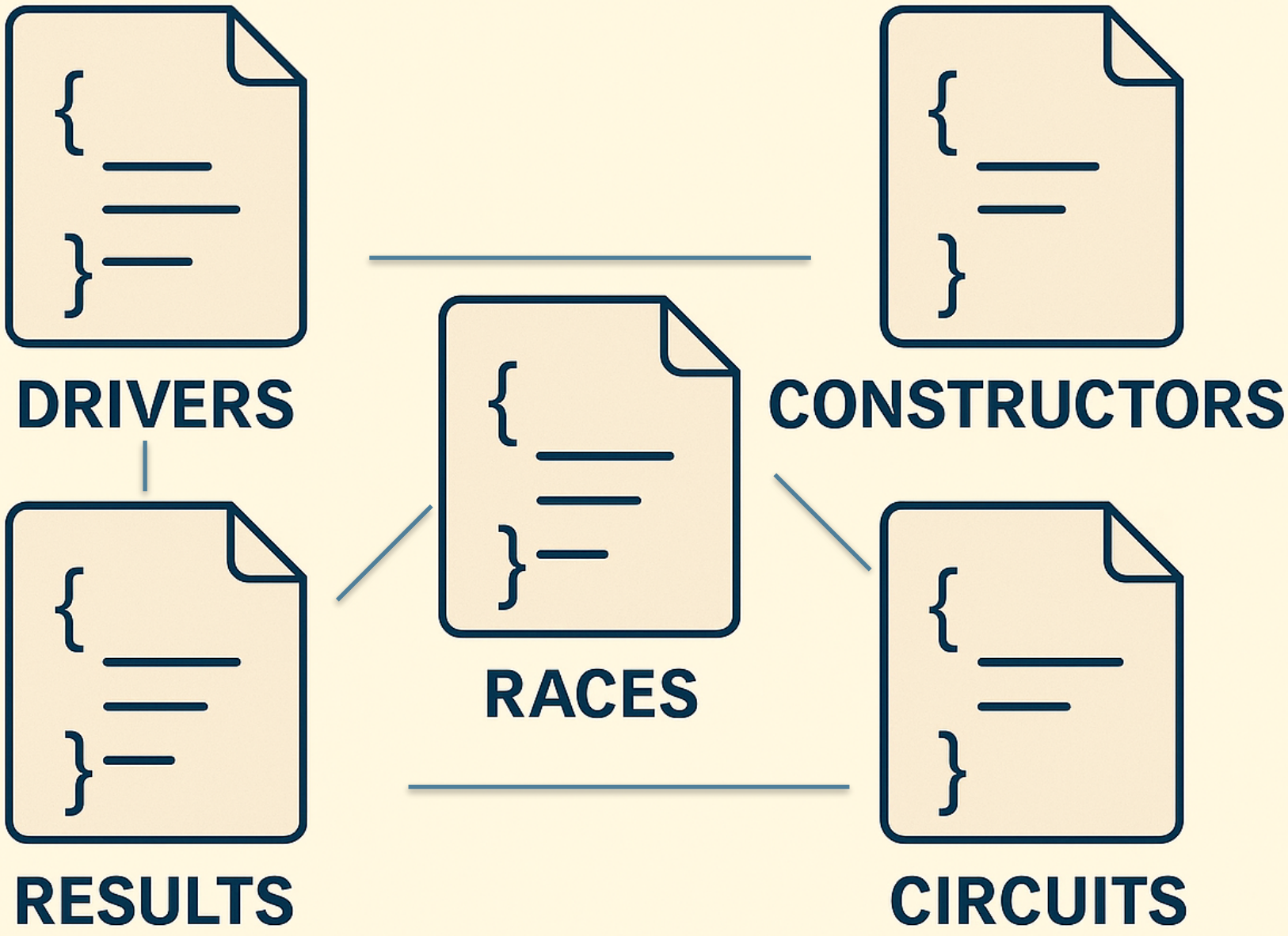


RESULTS



CIRCUITS


DRIVERS
CONSTRUCTORS
RACES
RACES
RESULTS
CIRCUITS


DRIVERS
CONSTRUCTORS
RACES
RESULTS
CIRCUITS

```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"]
           }'
        )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```

```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"]
           }'
        )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```

```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS"                                    ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV"                                    ]
           }'
        )
    );
  execute immediate schema_sql;
  dbms_json_duality.import(table_name => 'CIRCUITDOCS', view_name => 'CIRCUITS_DV');
end;
/
```

```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS", "CONSTRUCTORDOCS", "DRIVERDOCS", "RACEDOCS", "RESULTDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV", "CONSTRUCTORS_DV", "DRIVERS_DV", "RACES_DV", "RESULTS_DV" ]
           }'
        )
    );
  execute immediate schema_sql;
```

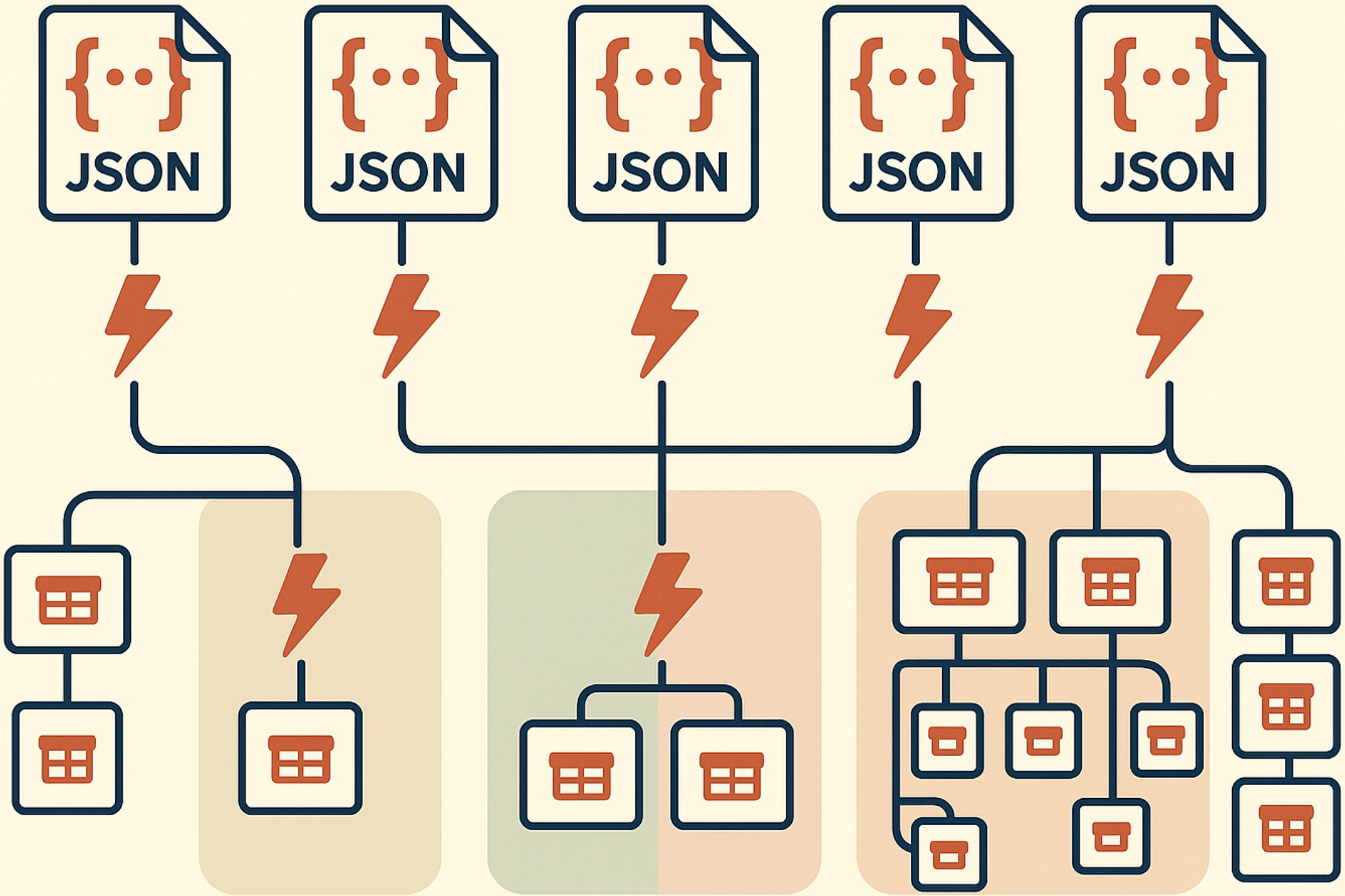
```
declare
  schema_sql clob;
begin
  schema_sql :=
    dbms_json_duality.infer_and_generate_schema(
    json( '{ "outputFormat"  : "executable"
           , "sourceSchema"  : "HOWTO"
           , "tableNames"    : [ "CIRCUITDOCS", "CONSTRUCTORDOCS", "DRIVERDOCS", "RACEDOCS", "RESULTDOCS" ]
           , "targetSchema"  : "HOWTO"
           , "updatability"  : true
           , "useFlexFields" : false
           , "viewNames"     : [ "CIRCUITS_DV", "CONSTRUCTORS_DV", "DRIVERS_DV", "RACES_DV", "RESULTS_DV" ]
           }'
        )
    );
  execute immediate schema_sql;
  dbms_errlog.create_error_log( dml_table_name => 'CIRCUITDOCS'
                              , err_log_table_name => 'ERR$_CIRCUITDOCS'
                              , skip_unsupported => TRUE);
  dbms_errlog.create_error_log( dml_table_name => 'CONSTRUCTORDOCS'
                              , err_log_table_name => 'ERR$_CONSTRUCTORDOCS'
                              , skip_unsupported => TRUE);
  dbms_errlog.create_error_log( dml_table_name => 'DRIVERDOCS'
                              , err_log_table_name => 'ERR$_DRIVERDOCS'
                              , skip_unsupported => TRUE);
```

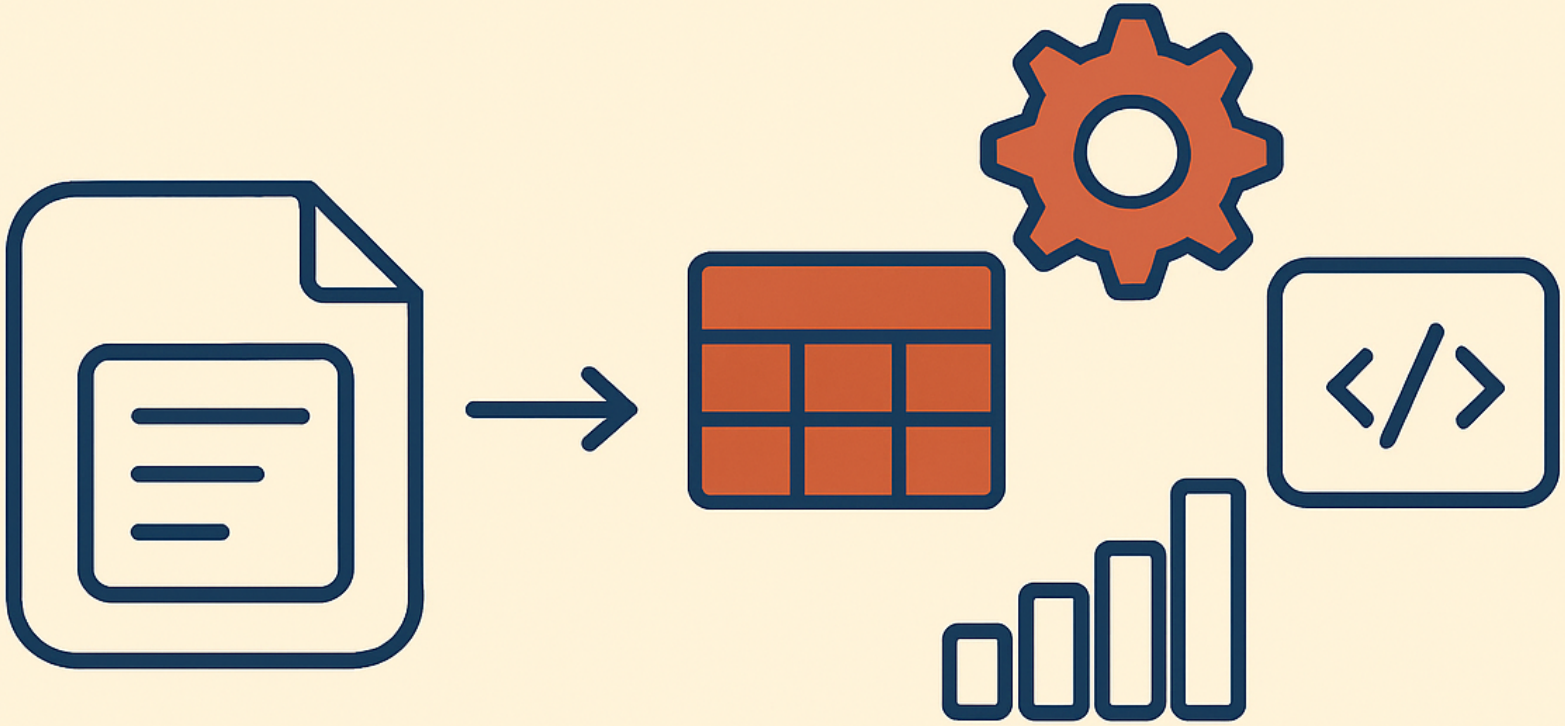
```
  sys.dbms_json_duality.import_all(
    json('{ "tableNames"    : [ "CIRCUITDOCS", "CONSTRUCTORDOCS", "DRIVERDOCS", "RACEDOCS", "RESULTDOCS" ]
          , "viewNames"     : [ "CIRCUITS_DV", "CONSTRUCTORS_DV", "DRIVERS_DV", "RACES_DV", "RESULTS_DV" ]
          , "sourceSchema"  : "HOWTO"
          , "targetSchema"  : "HOWTO"
          , "errorLog"      : [ "ERR$_CIRCUITDOCS", "ERR$_CONSTRUCTORDOCS", "ERR$_DRIVERDOCS"
                              , "ERR$_RACEDOCS", "ERR$_RESULTDOCS" ]
          }')
        );
end;
/
```

```
EXECUTE IMMEDIATE 'CREATE OR REPLACE JSON RELATIONAL DUALITY VIEW "HOWTO".RESULTS_DV AS
"HOWTO".results_dv_root @insert @update @delete
{
  _id : round
  url
  date: ora$date
  time
  round @generated (path: "$._id")
  season
  Circuit: "HOWTO".circuits_dv_root @insert @update @object @nocheck
  {
    url
    Location: "HOWTO".circuits_dv_root @insert @update @object @nocheck @link(from:
[circuit_id_circuits_dv_root])
    {
      lat
      long: ora$long
      country
      locality
      ora$id: circuit_id @nocheck
    }
    circuitId: circuit_id @nocheck
    circuitName: circuit_name
  }
  Results: "HOWTO".results_dv_results @insert @update @delete @array
  [{
    Time: "HOWTO".results_dv_time @insert @update @object
    {
      time
      millis
    }
    grid
    laps
    Driver: "HOWTO".drivers_dv_root @insert @update @object
```

```
grid
laps
Driver: "HOWTO".drivers_dv_root @insert @update @object
{
  url
  code
  driverId: driver_id
  givenName: given_name
  familyName: family_name
  dateOfBirth: date_of_birth
  nationality
  permanentNumber: permanent_number
}
number: ora$number
points
status
position
FastestLap: "HOWTO".results_dv_fastest_lap @insert @update @object
{
  lap
  Time: "HOWTO".results_dv_time2 @insert @update @object
  {
    time
  }
  rank
}
Constructor: "HOWTO".constructors_dv_root @insert @update @object
{
  url
  name
  nationality
  constructorId: constructor_id
}
positionText: position_text
```

```
    {
      url
      code
      driverId: driver_id
      givenName: given_name
      familyName: family_name
      dateOfBirth: date_of_birth
      nationality
      permanentNumber: permanent_number
    }
    number: ora$number
    points
    status
    position
    FastestLap: "HOWTO".results_dv_fastest_lap @insert @update @object
    {
      lap
      Time: "HOWTO".results_dv_time2 @insert @update @object
      {
        time
      }
      rank
    }
    Constructor: "HOWTO".constructors_dv_root @insert @update @object
    {
      url
      name
      nationality
      constructorId: constructor_id
    }
    positionText: position_text
  }]
  raceName: race_name
}';
```


JSON
JSON
JSON
JSON
JSON



THIS IS NOT AN END RESULT BUT A STARTING POINT

Q&A

NL.OUG
NL ORACLE USER GROUP

How to use a relational database for your JSON documents-Patrick Barel

Please fill in your evaluations