

Unleashing the power of

# JSON RELATIONAL DUALITY VIEWS

in APEX

Kevin Thyssen

March 20th · Ede · The Netherlands



 Oracle ACE  
Associate

Senior APEX consultant @ United Codes

Located in Belgium

15+ years experience in Oracle APEX

Cycling enthusiast



@kevinthyssen



kevin.thyssen@united-codes.com



kevinthyssen

# Extend APEX like a PRO



**APEX Office Print**  
Printing & Reporting  
built for APEX



**APEX Media Extension**  
Image processing for the  
Oracle Database.



**APEX Office Edit**  
Document management  
for APEX



**Plug-ins Pro**  
Document management  
for APEX

**APEX Message Service**  
Real-time communication  
for Oracle APEX



**APEX Project Eye**  
Productivity, Quality Assurance,  
and Security tool for APEX

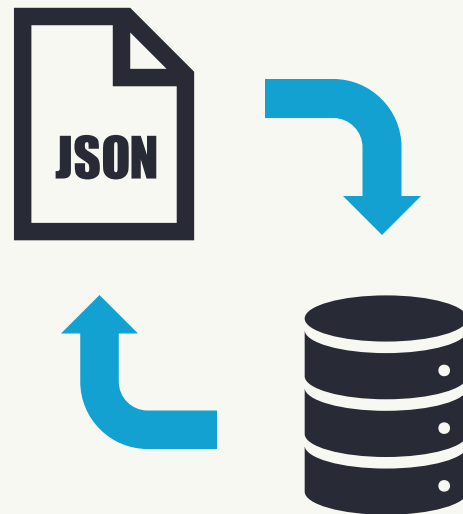


# JSON RELATIONAL DUALITY VIEWS

## Introduction

### Concept of JSON Relational Duality

- Data is organized both **relationally** and **hierarchically**
- A **JSON Relational Duality View** exposes data stored in relational database tables as JSON documents
- Applications can access and modify the same data as a set of JSON documents or as a set of related tables and columns, and both approaches can be employed at the same time



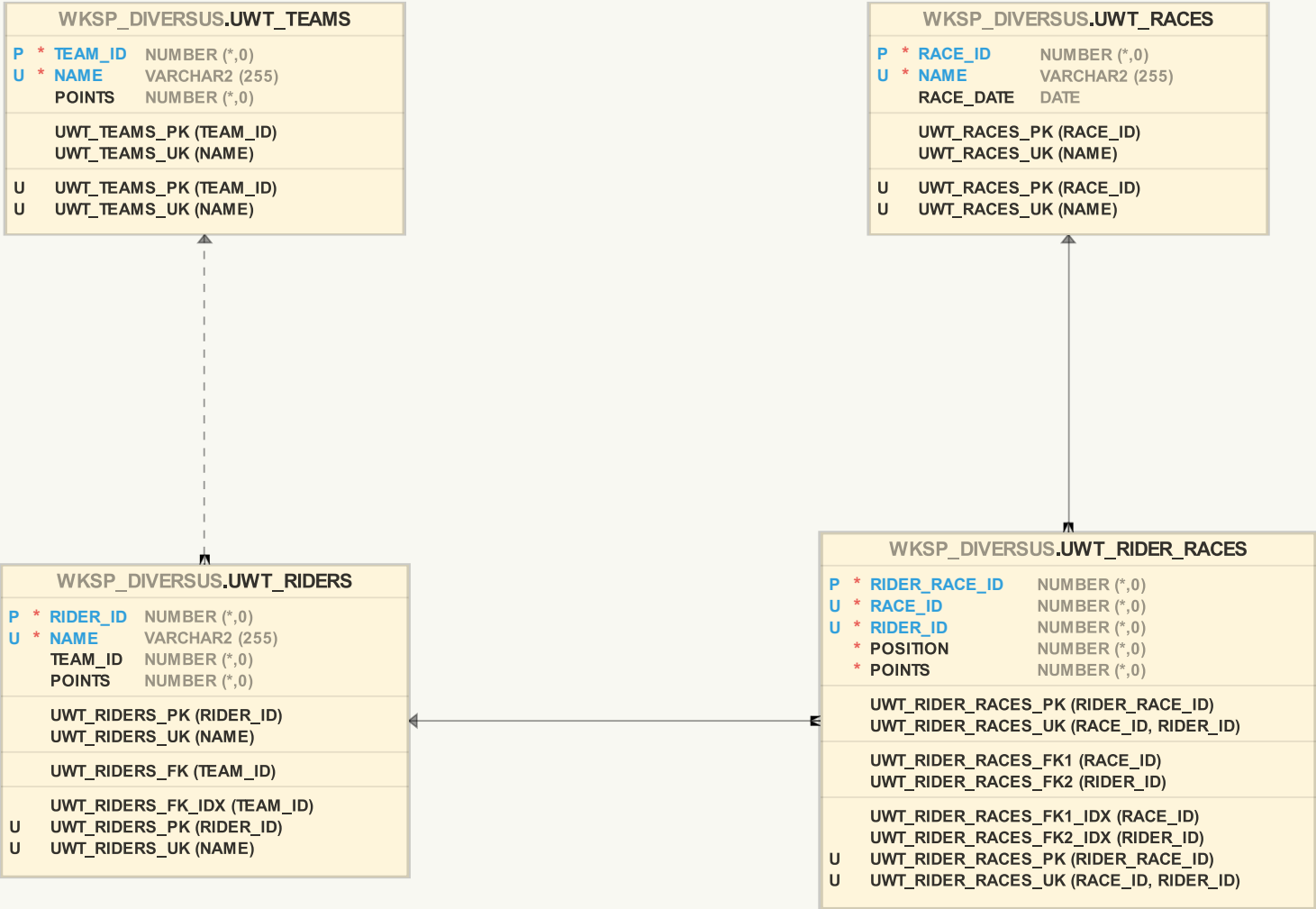
## Introduction

### Short facts: A few things you should know

- Available in 23ai
- Data is not stored, but generated on demand
- Read-only by default, partially or completely updatable
- All views have one DATA column of data type JSON
- SQL or GraphQL syntax
- Advantages:
  - JSON: Straightforward, common interchange format
  - Relational: Consistency, space efficiency, normalization

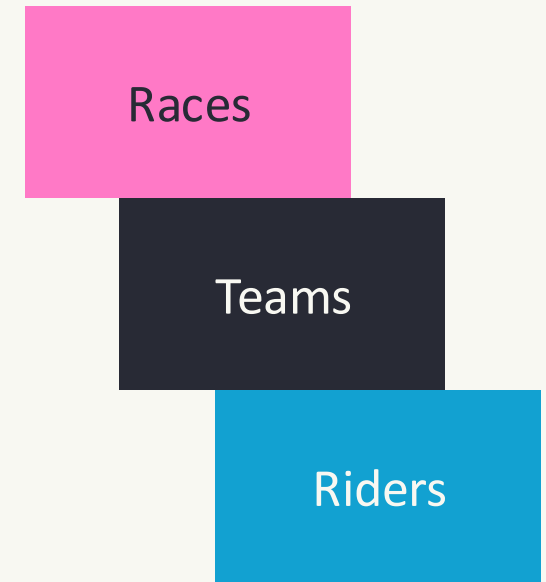
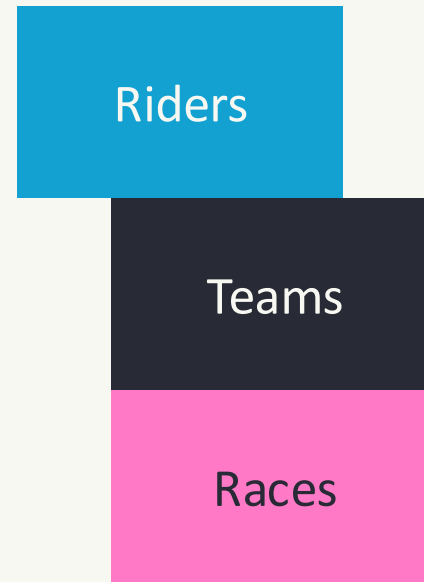
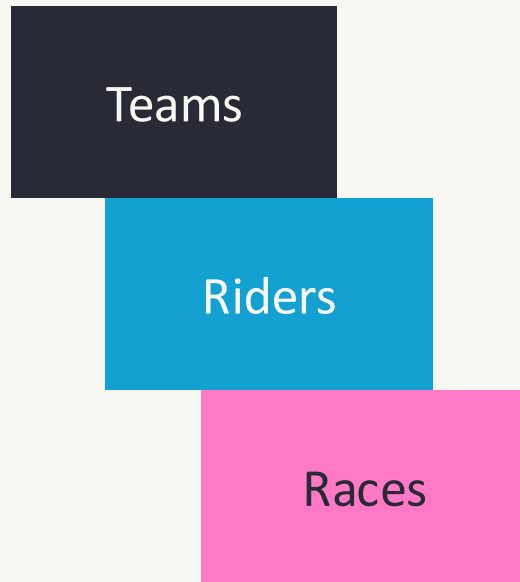
# Introduction

## UCI World Tour: Example data model



# Introduction

## UCI World Tour: JSON hierarchies



# Introduction

## UCI World Tour: Sample application

≡ JSON Duality Views

🔍 kevin ▾

Q ▾

Go

Actions ▾

Create

|                                                                                     | Rider ID | Name                 | Team                      | Races | Points |
|-------------------------------------------------------------------------------------|----------|----------------------|---------------------------|-------|--------|
|    | 12       | BENOOT Tiesj         | Team Visma   Lease A Bike | 1     | 520    |
|    | 5        | BETTIOL Alberto      | EF Education - EasyPost   | 1     | 360    |
|    | 9        | BJERG Mikkel         | UAE Team Emirates         | 1     | 440    |
|    | 2        | MATTHEWS Michael     | Team Jayco AlUla          | 1     | 640    |
|    | 10       | MORGADO António      | UAE Team Emirates         | 1     | 360    |
|    | 7        | MOZZATO Luca         | Arkéa - B&B Hotels        | 1     | 640    |
|    | 4        | PEDERSEN Mads        | Lidl - Trek               | 1     | 440    |
|    | 1        | PHILIPSEN Jasper     | Alpecin - Deceuninck      | 1     | 800    |
|    | 3        | POGAČAR Tadej        | UAE Team Emirates         | 1     | 520    |
|    | 8        | POLITT Nils          | UAE Team Emirates         | 1     | 520    |
|  | 11       | VAN AERT Wout        | Team Visma   Lease A Bike | 0     | 0      |
|  | 6        | VAN DER POEL Mathieu | Alpecin - Deceuninck      | 1     | 800    |

1 - 12

# Example 1 - Teams

## Definition & annotations

```
create json relational duality view uwt_teams_dv as
select json {
  '_id' : t.team_id,
  'name' : t.name,
  'points' : t.points,
  'riders' :
    [ select json {
      'riderId' : ri.rider_id,
      'name' : ri.name,
      'points' : ri.points }
      from uwt_riders ri with insert update
      where ri.team_id = t.team_id ] }
from uwt_teams t with insert update delete;
```

# Example 1 - Teams

## Definition & annotations

```
create json relational duality view uwt_teams_dv as
select json {
  '_id' : t.team_id,
  'name' : t.name,
  'points' : t.points,
  'riders' :
    [ select json {
      'riderId' : ri.rider_id,
      'name' : ri.name,
      'points' : ri.points }
      from uwt_riders ri with insert update
      where ri.team_id = t.team_id ] }
from uwt_teams t with insert update delete;
```

```
{
  "_id": 1,
  "name": "Alpecin - Deceuninck",
  "points": 1600,
  "riders": [
    {
      "riderId": 1,
      "name": "PHILIPSEN Jasper",
      "points": 800
    },
    {
      "riderId": 6,
      "name": "VAN DER POEL Mathieu",
      "points": 800
    }
  ],
  "_metadata": {
    "etag": "C9FC18A4BC7D50173A1BDD52D7A31E9A",
    "asof": "000025BA12AB9FB8"
  }
}
```

## Example 1 - Teams

### Definition & annotations

```
create json relational duality view uwt_teams_dv as
select json {
  '_id' : t.team_id,
  'name' : t.name,
  'points' : t.points,
  'riders' :
    [ select json {
      'riderId' : ri.rider_id,
      'name' : ri.name,
      'points' : ri.points }
      from uwt_riders ri with insert update
      where ri.team_id = t.team_id ] }
from uwt_teams t with insert update delete;
```

- Document-identifier field that corresponds to the root-table primary-key column(s) of the duality view
- Can be an object for composite primary keys (multiple columns) or to provide a more meaningful name

```
{
  "_id": {
    "teamId": 1,
    "name": "Alpecin - Deceuninck"
  },
  ...
}
```

## Example 1 - Teams

### Definition & annotations

```
create json relational duality view uwt_teams_dv as
  select json {
    '_id' : t.team_id,
    'name' : t.name,
    'points' : t.points,
    'riders' :
      [ select json {
        'riderId' : ri.rider_id,
        'name' : ri.name,
        'points' : ri.points }
        from uwt_riders ri with insert update      with insert update
        where ri.team_id = t.team_id ] }
  from uwt_teams t with insert with insert update delete
```

- Duality views are read-only by default: with noinsert nouupdate nodelete (implicit)
- Annotations on table-level and column-level (column-level overrides table-level)
- Primary-key columns always read-only
-  Think carefully about which annotations you want to allow

## Example 1 - Teams

### Metadata

- **etag**: A hash value of the document content, used for optimistic concurrency control
- **asof**: System Change Number of the document, internal database time stamp

```
{
  "_id": 1,
  "name": "Alpecin - Deceuninck",
  "points": 1600,
  "riders": [
    {
      "riderId": 1,
      "name": "PHILIPSEN Jasper",
      "points": 800
    },
    {
      "riderId": 6,
      "name": "VAN DER POEL Mathieu",
      "points": 800
    }
  ],
  "_metadata": {
    "etag": "C9FC18A4BC7D50173A1BDD52D7A31E9A",
    "asof": "000025BA12AB9FB8"
  }
}
```

## Example 1 - Teams

### GraphQL

```
create json relational duality view uwt_teams_dv as
uwt_teams @insert @update @delete {
  _id : team_id @noupdate
  name
  points
  uwt_riders @insert @update
    @link (to : [team_id]) {
      riderid : rider_id @noupdate
      name
      points
    }
};
```

```
{
  "_id": 1,
  "name": "Alpecin - Deceuninck",
  "points": 1600,
  "riders": [
    {
      "riderId": 1,
      "name": "PHILIPSEN Jasper",
      "points": 800
    },
    {
      "riderId": 6,
      "name": "VAN DER POEL Mathieu",
      "points": 800
    }
  ],
  "_metadata": {
    "etag": "C9FC18A4BC7D50173A1BDD52D7A31E9A",
    "asof": "000025BA12AB9FB8"
  }
}
```

## Example 1 - Teams

### AutoREST

- REST enable database objects like tables and views
- GET (read), POST (insert), PUT (update) & DELETE (delete)
- New since ORDS 23.2: PATCH (partial update)
- Batch Load

APEX App Builder SQL Workshop Team Development Gallery

RESTful Services \ Enabled Objects Schema WKSP\_DIVERSUS

RESTful Data Services

- Enabled Objects
- Modules
  - riders
    - /
    - GET
    - POST
    - :id
    - DELETE
    - GET
    - PUT

Enabled Objects

Search Go Actions Create AutoREST Object >

| Parsing Object | Object Alias  | Type | Status  | AutoREST Authorization | Ops Allowed | Type Path | Pre Hook | Aliased |
|----------------|---------------|------|---------|------------------------|-------------|-----------|----------|---------|
| UWT_TEAMS_DV   | uw_t_teams_dv | View | ENABLED | DISABLED               |             | ENABLED   |          |         |

1 - 1

Legend: Object name and alias are different Object name and alias are the same

## Example 1 - Teams

### AutoREST - What checks does a duality view enforce?

- Basic data type checking
- Not null columns
- Check constraints & unique constraints
- Referential integrity
  
- But: the database is not aware of any business logic in APEX or PL/SQL packages

## Example 2 - Riders

### Definition & annotations

```
create json relational duality view uwt_riders_dv as
select json {
  '_id' : ri.rider_id,
  'name' : ri.name,
  'points' : ri.points with nocheck,
  'teamInfo' :
    ( select json {
      'teamId' : t.team_id,
      'name' : t.name }
      from uwt_teams t with insert update
      where t.team_id = ri.team_id ),
  'races' :
    [ select json {
      'riderRaceId' : rr.rider_race_id,
      'position' : rr.position,
      'points' : rr.points,
      unnest
        ( select json {
          'raceId' : ra.race_id,
          'name' : ra.name }
          from uwt_races ra
          where ra.race_id = rr.race_id ) }
      from uwt_rider_races rr with insert update delete
      where rr.rider_id = ri.rider_id ] }
from uwt_riders ri with insert update delete;
```

## Example 2 - Riders

### Definition & annotations

```
create json relational duality view uwt_riders_dv as
select json {
  '_id' : ri.rider_id,
  'name' : ri.name,
  'points' : ri.points with nocheck,
  'teamInfo' :
    ( select json {
      'teamId' : t.team_id,
      'name' : t.name }
      from uwt_teams t with insert update
      where t.team_id = ri.team_id ),
  'races' :
    [ select json {
      'riderRaceId' : rr.rider_race_id,
      'position' : rr.position,
      'points' : rr.points,
      unnest
        ( select json {
          'raceId' : ra.race_id,
          'name' : ra.name }
          from uwt_races ra
          where ra.race_id = rr.race_id ) }
      from uwt_rider_races rr with insert update delete
      where rr.rider_id = ri.rider_id ] }
from uwt_riders ri with insert update delete;
```

```
{
  "_id": 6,
  "name": "VAN DER POEL Mathieu",
  "points": 800,
  "teamInfo": {
    "teamId": 1,
    "name": "Alpecin - Deceuninck"
  },
  "races": [
    {
      "riderRaceId": 6,
      "position": 1,
      "points": 800,
      "raceId": 2,
      "name": "Ronde van Vlaanderen"
    }
  ],
  "_metadata": {
    "etag": "1CDCB8965BCE626F603F7B5378D2B6E5",
    "asof": "000025BA12E39AD3"
  }
}
```

## Example 2 - Riders

### Definition & annotations

```
create json relational duality view uwt_riders_dv as
select json {
  '_id' : ri.rider_id,
  'name' : ri.name,
  'points' : ri.points with nocheck,      with nocheck
  'teamInfo' :
    ( select json {
      'teamId' : t.team_id,
      'name' : t.name }
      from uwt_teams t with insert update
      where t.team_id = ri.team_id ),
  'races' :
    [ select json {
      'riderRaceId' : rr.rider_race_id,
      'position' : rr.position,
      'points' : rr.points,
      unnest
        ( select json {
          'raceId' : ra.race_id,
          'name' : ra.name }
          from uwt_races ra
          where ra.race_id = rr.race_id ) }
      from uwt_rider_races rr with insert update delete
      where rr.rider_id = ri.rider_id ] }
from uwt_riders ri with insert update delete;
```

- By default, all fields of a document contribute to the calculation of the value of field **etag**
- **nocheck**: Exclude given field from calculation
- Annotations on table-level and column-level (column-level overrides table-level)
- Primary-key columns always have the default behavior of contributing to **etag** calculation, unless an explicit column-level annotation of **nocheck** is given

## Example 2 - Riders

### Definition & annotations

```
create json relational duality view uwt_riders_dv as
select json {
  '_id' : ri.rider_id,
  'name' : ri.name,
  'points' : ri.points with nocheck,
  'teamInfo' :
    ( select json {
      'teamId' : t.team_id,
      'name' : t.name }
      from uwt_teams t with insert update
      where t.team_id = ri.team_id ),
  'races' :
    [ select json {
      'riderRaceId' : rr.rider_race_id,
      'position' : rr.position,
      'points' : rr.points,
      unnest      unnest
        ( select json {
          'raceId' : ra.race_id,
          'name' : ra.name }
          from uwt_races ra
          where ra.race_id = rr.race_id ) }
      from uwt_rider_races rr with insert update delete
      where rr.rider_id = ri.rider_id ] }
from uwt_riders ri with insert update delete;
```

- Flattens that nested object to just include its fields directly

## Example 2 - Riders

### RESTful Services

The screenshot displays the APEX SQL Workshop interface. The top navigation bar includes the APEX logo, 'App Builder', 'SQL Workshop', 'Team Development', and 'Gallery' tabs. A search bar and user profile 'KE kevin diversus' are also present. The breadcrumb trail indicates the current location: RESTful Services > Modules > Module Definition > Resource Template > Resource Handler. The left-hand navigation pane shows a tree structure under 'RESTful Data Services', with 'Modules' expanded to show 'riders'. Under 'riders', the resource path '/' is selected, and the 'POST' method is highlighted. The right-hand pane, titled 'Resource Handler', contains the following configuration:

- RESTful Service Module: **riders**
- Module Base Path: **/riders/**
- URI Template: **/**
- Full URL: <https://ofxrel3koy69ptc-db23ai.adb.eu-frankfurt-1.oraclecloudapps.com/ords/diversus/riders//>
- Method: **POST**
- Source Type: **PL/SQL**
- Mime Types Allowed: (empty field)
- Comments: (empty text area)

Buttons for 'Cancel', 'Delete', and 'Apply Changes' are located at the top right of the Resource Handler pane. Below the configuration, the 'Source' pane shows the PL/SQL code for the POST method:

```
3  l_json  clob;
4  begin
5      insert into uwt_riders_dv ri (data)
6      values (:body_text)
7      returning ri.data."_id" into l_rider_id;
8
9      uwt_pkg.upd_race_points;
10
11     select ri.data
12     into l_json
13     from uwt_riders_dv ri
14     where ri.data."_id" = l_rider_id;
15
16     apex_util.prn(p_clob => l_json,
17                 p_escape => false);
18 end;
```

# SQL Developer for VS Code

## Duality View Builder

The screenshot displays the SQL Developer for VS Code interface, specifically the Duality View Builder for a connection named 'OCI - DB23AI - DIVERSUS'. The main workspace shows a diagram of two tables: 'UWT\_TEAMS' and 'UWT\_RIDERS', connected by a foreign key relationship 'UWT\_RIDERS\_FK'. The 'UWT\_TEAMS' table has columns: TEAM\_ID (NUMBER(\*,0)), NAME (VARCHAR2(255)), and POINTS (NUMBER(\*,0)). The 'UWT\_RIDERS' table has columns: RIDER\_ID (NUMBER(\*,0)), NAME (VARCHAR2(255)), TEAM\_ID (NUMBER(\*,0)), and POINTS (NUMBER(\*,0)).

On the right side, the 'UWT\_RIDERS' table is selected, and its columns are listed: RIDER\_ID, NAME, and POINTS. Below this, the 'Alias' section is visible, showing options for 'Check', 'Allow Update', 'Allow Insert', and 'Allow Delete'. The 'Table Filter' section is also present, with an 'Apply' button.

At the bottom, the 'Definition' tab is active, showing the SQL definition for the 'UWT\_TEAMS\_DV' view:

```
1 CREATE FORCE EDITIONABLE JSON RELATIONAL DUALITY VIEW uwt_teams_dv
2 AS
3 UWT_TEAMS @insert @update @delete {
4   _id : team_id @noupdate
5   name
6   points
7   UWT_RIDERS @insert @update
8     @link (to : [TEAM_ID]) {
9       riderId : rider_id @noupdate
10      name
11      points
12    }
13 }
```

The bottom status bar shows the 'SQL SNIPPETS' section with a 'Sign in to Jira' button and a 'No active issue' message. The bottom right corner displays the 'Go Live' button and the 'Supervision' status.

**THANK YOU!**