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 enthousiast

@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

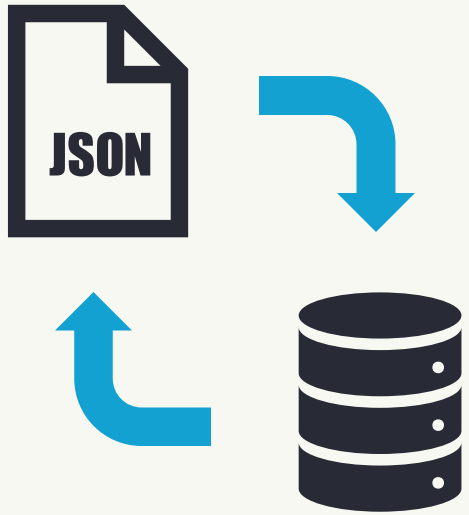
APEX Message Service
Real-time communication
for Oracle APEX

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

JSON RELATIONAL DUALITY VIEWS

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

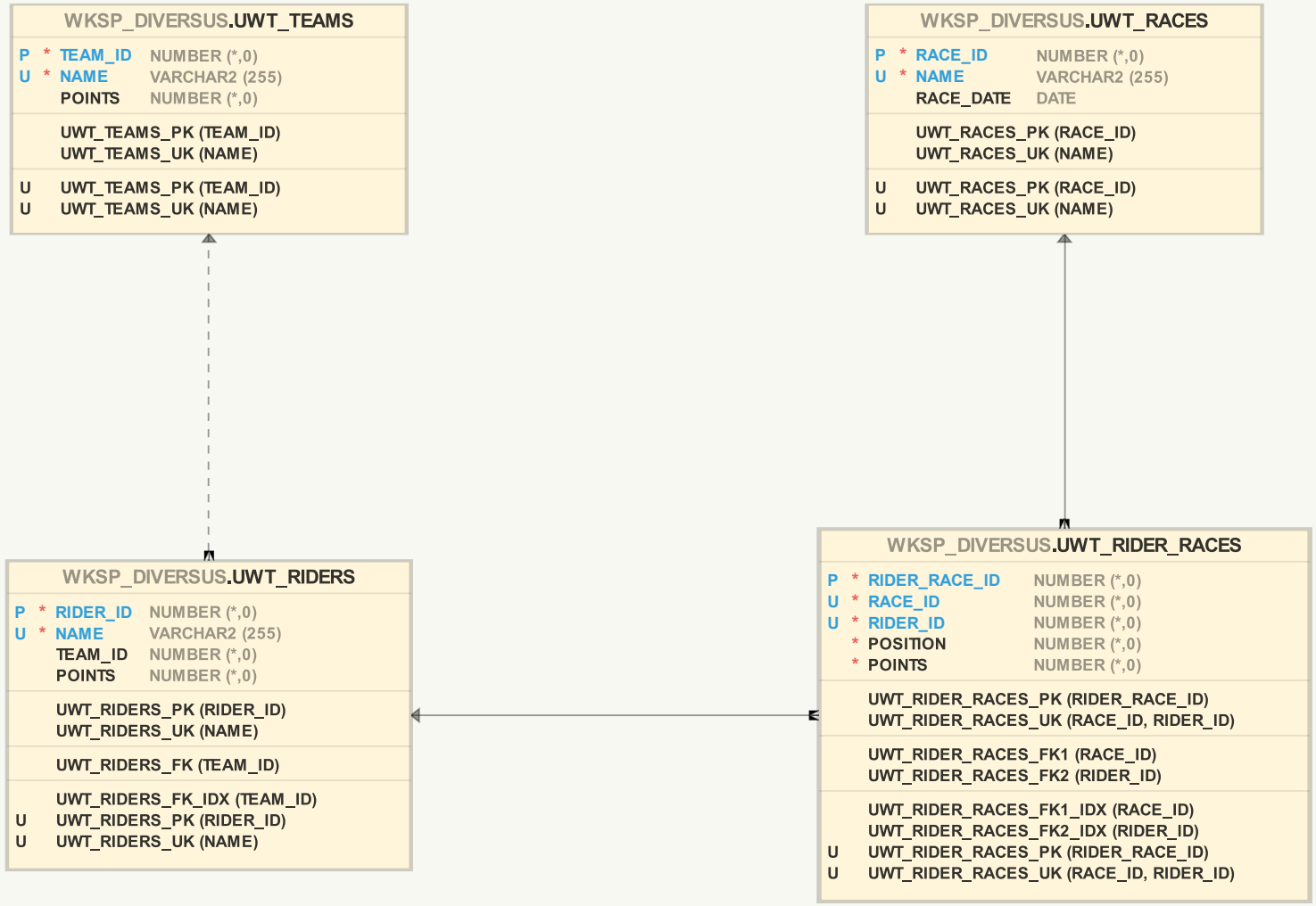


Short facts: A few things you should know

- 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

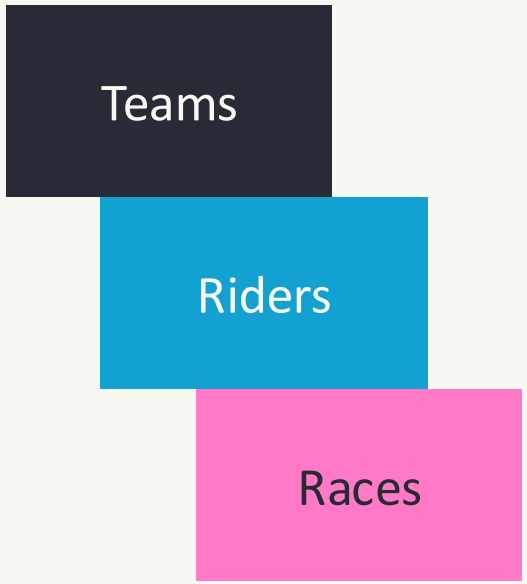
Introduction

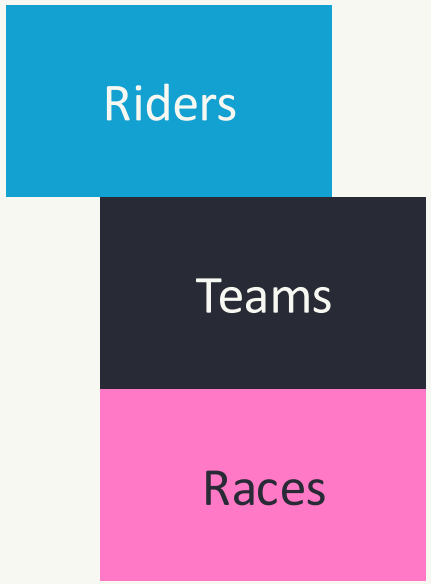
UCI World Tour: Example data model

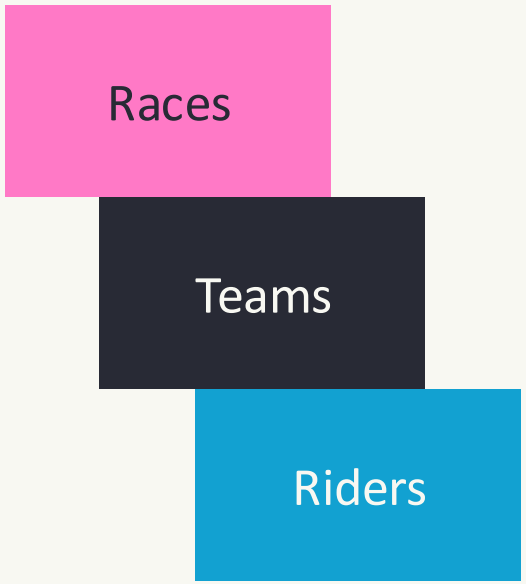


Introduction

UCI World Tour: JSON hierarchies

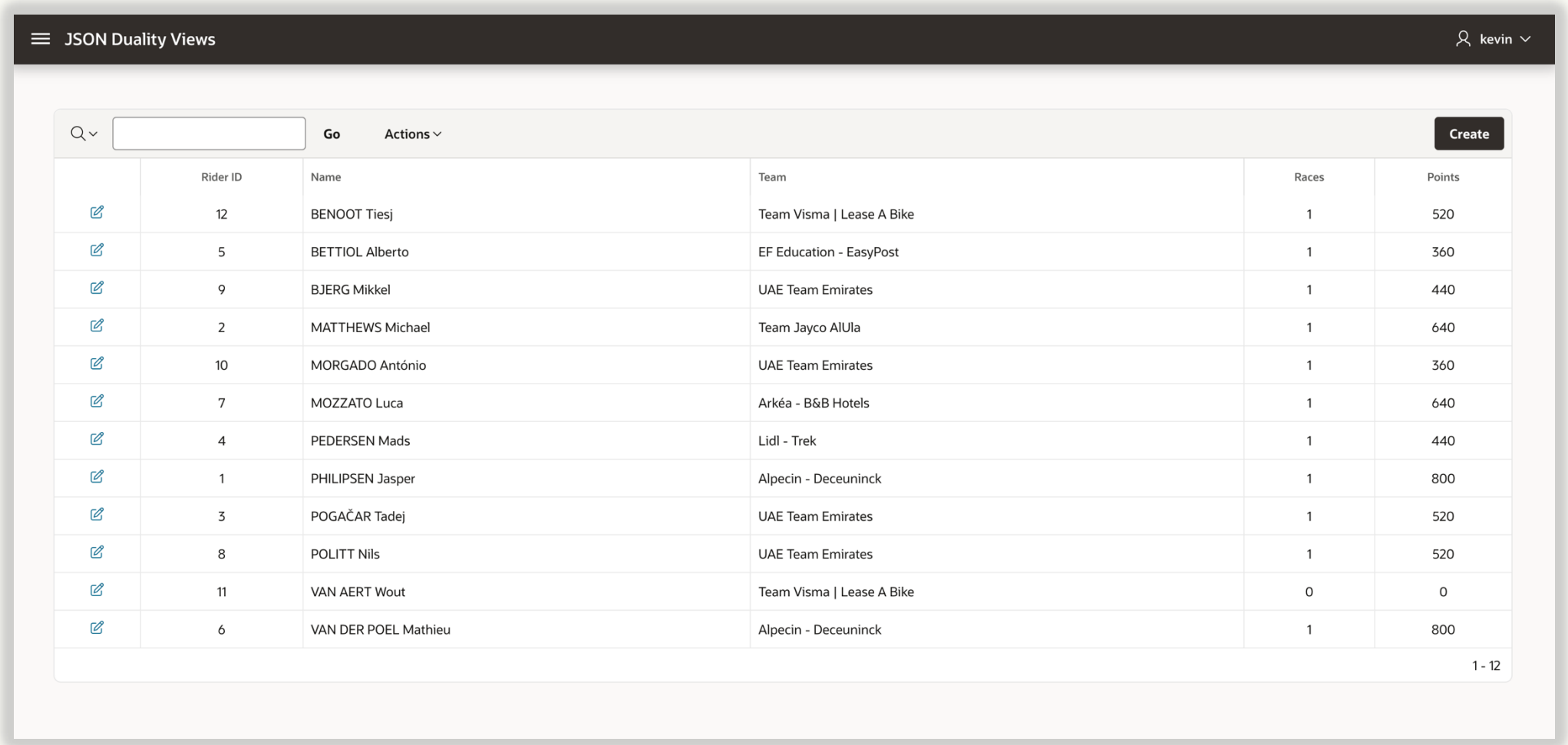






Introduction

UCI World Tour: Sample application



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
           where ri.team_id = t.team_id ] }
    from uwt_teams t with insert update delete
```

- Duality views are read-only by default: with noinsert noupdate 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

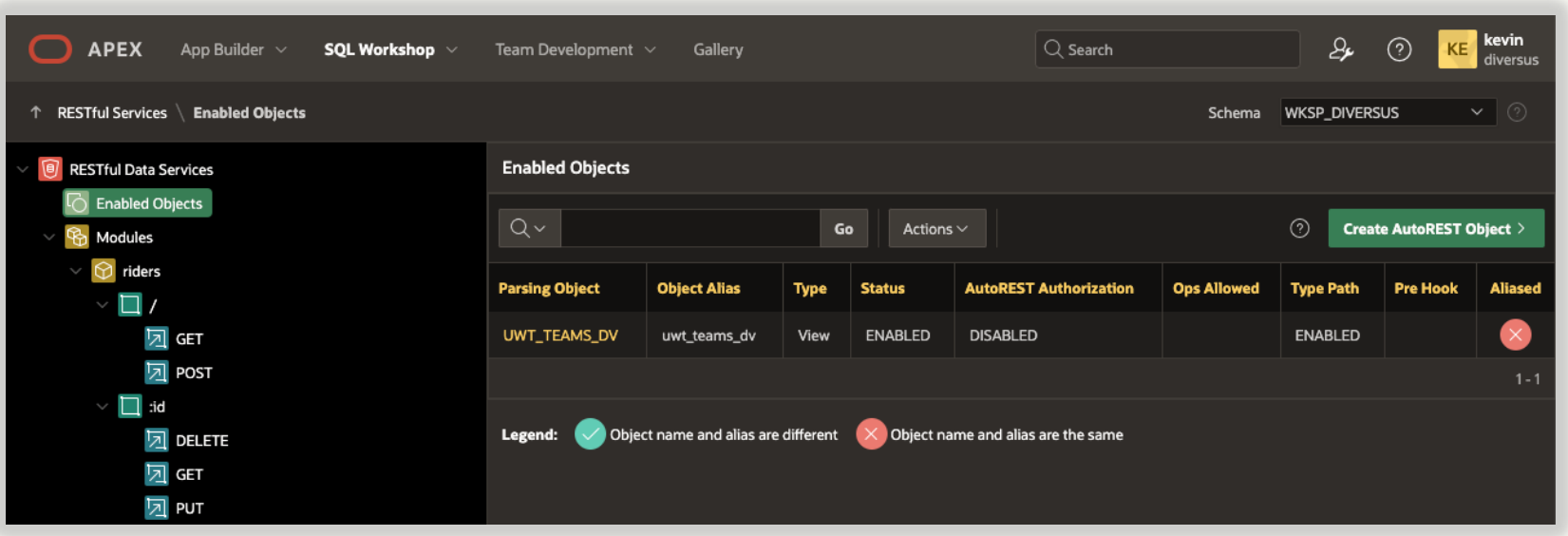
```
create json relational duality view uwt_teams_dv as
uwt_teams @insert @update @delete {
   _id : team_id @noupdate
   name
   points
   uwt_riders @insert @update
         @link (to : [team_id]) {
      riderid : rider_id @noupdate
      name
      points
   }
};
```


```
{
  "_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



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

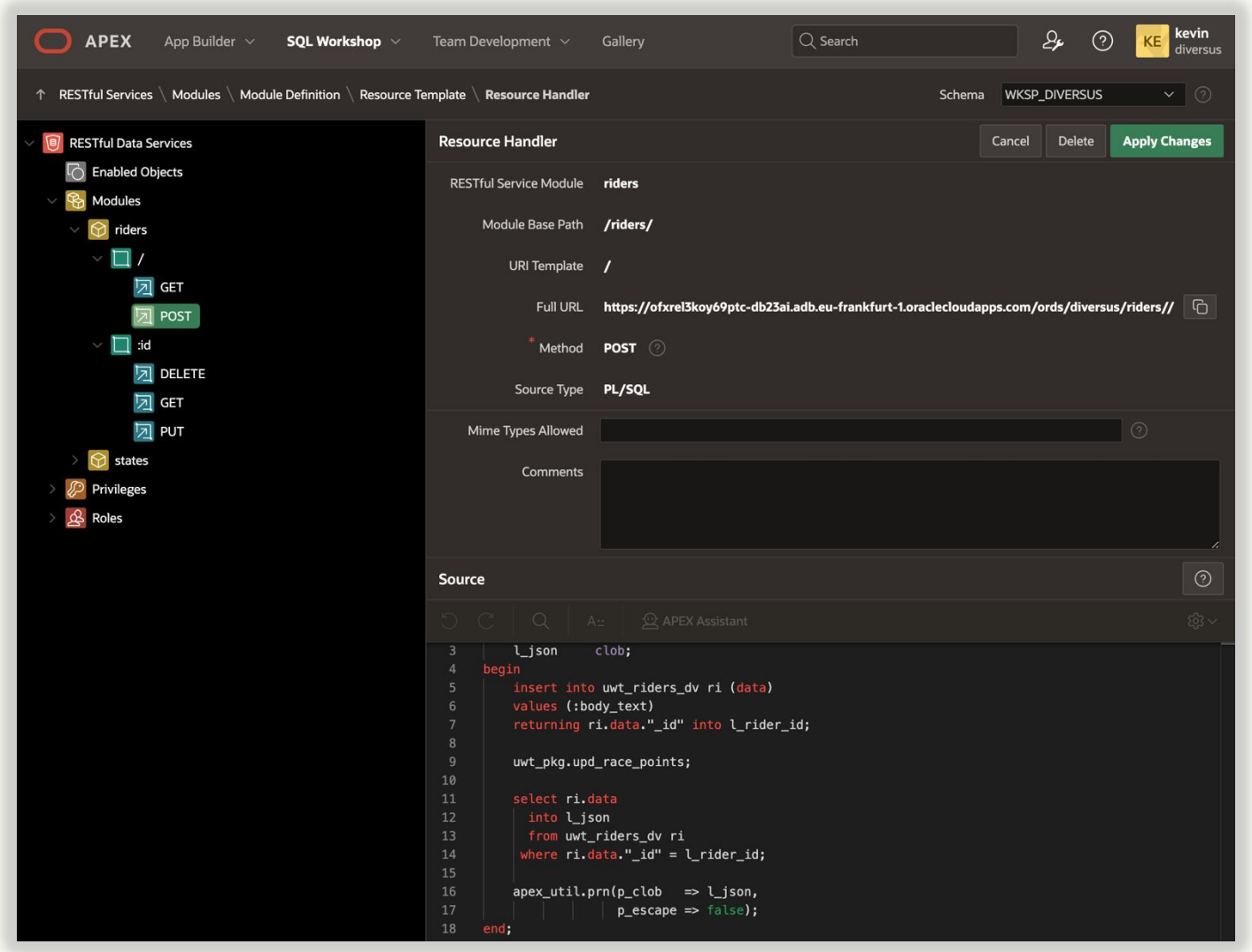
Example 2 - Riders
Definition & annotations

```
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

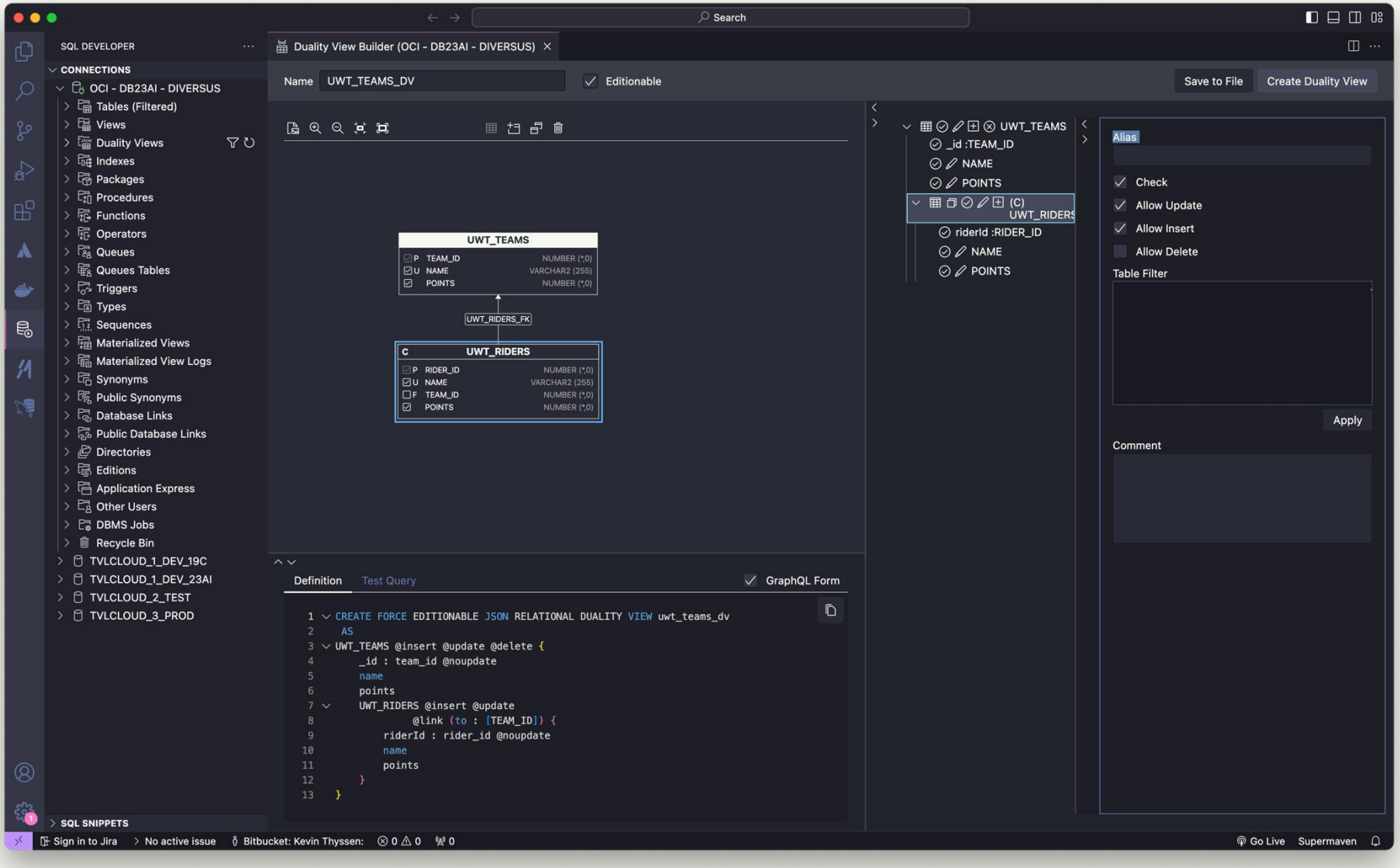
Example 2 - Riders

RESTful Services



SQL Developer for VS Code

Duality View Builder



THANK YOU!