6
Document-Identifier Field for Duality Views

A document supported by a duality view always includes, at its top level, a **document-identifier** field, **`_id`**, which corresponds to the *identifying columns* (primary-key columns, identity columns, or columns with a unique constraint or unique index) of the *root* table underlying the view. The field value can take different forms.

(An **identity column** is one whose numeric value is generated automatically and uniquely for each table row. You declare it using keywords `GENERATED BY DEFAULT ON NULL AS IDENTITY`.)

Often there is only one such identifying column and it is often a primary-key column. If there is more than one primary-key column then we sometimes speak of the primary key being **composite**.

- If there is only *one identifying column* then you use that as the value of field `_id` when you define the duality view.
- Alternatively, you can use an object as the value of field `_id`. The members of the object specify fields whose values are the identifying columns. An error is raised if there is not a field for each of the identifying columns.

  If there is only one identifying column, you can nevertheless use an object value for `_id`; doing so lets you provide a meaningful field name.

> **Note:**
>
> A duality view must have an `_id` field at its top level, to uniquely identify a given row of its root table, and thus the corresponding document.
>
> In order to *replicate* a duality view, you also need to ensure that each document *subobject*[1] has a top-level field whose value is the identifying columns for that table. That is, the columns corresponding to such a table **row-identifier** field need to uniquely identify a row of the table that underlies that subobject.
>
> Just as for a document-identifer field, the columns corresponding to a row-identifer field can be primary-key columns, identity columns, or columns with a unique constraint or unique index, for their table.
>
> A document-identifier field must be named `_id`. A row-identifier field can have any name, but if its name is `_id` then it's up to *you to ensure* that the corresponding columns uniquely identify a table row.

Example 6-1 Document Identifier Field _id With Primary-Key Column Value

For duality view `race_dv`, the value of a single primary-key column, `race_id`, is used as the value of field `_id`. A document supported by the view would look like this: `{"_id" : 1,…}`.

[1] This of course doesn't apply to document subobjects that are explicitly present a `JSON`-type column that's embedded in the document.

GraphQL:

```
CREATE JSON RELATIONAL DUALITY VIEW race_dv AS
  race {_id     : race_id
        name    : name
        laps    : laps @NOUPDATE
        date    : race_date
        podium  : podium @NOCHECK,
        result  : ...};
```

SQL:

```
CREATE JSON RELATIONAL DUALITY VIEW race_dv AS
  SELECT JSON {'_id'    : r.race_id,
               'name'   : r.name,
               'laps'   : r.laps WITH NOUPDATE,
               'date'   : r.race_date,
               'podium' : r.podium WITH NOCHECK,
               'result' : ...}
  FROM race r;
```

Example 6-2 Document Identifier Field _id With Object Value

For duality view `race_dv`, the value of field `_id` is an object with a single member, which maps the single primary-key column, `race_id`, to a meaningful field name, `raceId`. A document supported by the view would look like this: `{"_id" : {"raceId" : 1},...}`.

GraphQL:

```
CREATE JSON RELATIONAL DUALITY VIEW race_dv AS
  race {_id @nest {race_id}
        name    : name
        laps    : laps @NOUPDATE
        date    : race_date
        podium  : podium @NOCHECK,
        result  : ...};
```

SQL:

```
CREATE JSON RELATIONAL DUALITY VIEW race_dv AS
  SELECT JSON {'_id'    : {'raceId' : r.race_id},
               'name'   : r.name,
               'laps'   : r.laps WITH NOUPDATE,
               'date'   : r.race_date,
               'podium' : r.podium WITH NOCHECK,
               'result' : ...}
  FROM race r;
```

An *alternative* car-racing design might instead use a `race` table that has *multiple* identifying columns, `race_id` and `date`, which together identify a row. In that case, a document supported by the view would look like this: `{"_id" : {"raceId" : 1, "date" : "2022-03-20T00:00:00"},...}`.

GraphQL:

```
CREATE JSON RELATIONAL DUALITY VIEW race_dv AS
  race {_id @nest {raceId: race_id, date: race_date}
        name    : name
        laps    : laps @NOUPDATE
        podium  : podium @NOCHECK,
        result  : ...};
```

SQL:

```
CREATE JSON RELATIONAL DUALITY VIEW race_dv AS
  SELECT JSON {'_id'    : {'raceId' : r.race_id, 'date' : r.race_date},
               'name'   : r.name,
               'laps'   : r.laps WITH NOUPDATE,
               'podium' : r.podium WITH NOCHECK,
               'result' : ...}
  FROM race r;
```

Related Topics

- Car-Racing Example, JSON Documents
  The car-racing example has three kinds of documents: a team document, a driver document, and a race document.

> **See Also:**
>
> Mongo DB API Collections Supported by JSON-Relational Duality Views
>
> in *Oracle Database API for MongoDB*