
ORACLE
ACADEMY

Database Programming with SQL

15-1
Creating Views

ORACLE ACADEMY

Objectives

This lesson covers the following objectives:

- List three uses for views from the standpoint of a database administrator
- Explain, from a business perspective, why it is important to be able to create and use logical subsets of data derived from one or more tables
- Create a view with and without column aliases in the subquery using a single base table

Objectives

This lesson covers the following objectives:

- Create a complex view that contains group functions to display values from two tables
- Retrieve data from a view

Purpose

- Take a minute to look back at what you've learned so far as an Oracle Academy student.
- How easy would it be to explain what you know to someone who hasn't taken this class?
- You should pat yourself on the back!
- The level of knowledge you have acquired is understood by only a select few.

Purpose

- Now, imagine yourself as the Database Administrator of a business.
- What do you do when a manager asks you to make it possible for him to be able to retrieve and input data using the company's database?
- "Don't make it too complicated. I just want to be able to prepare reports about all our operations."

Purpose

- Should these employees have access to all of the company's data?
- How will they execute commands that require join conditions?
- Is it wise to allow data input from anyone?
- These are questions that you, as DBA, need to know how to answer.
- In this section, you will learn how to create "views" -- virtual representations of tables customized to meet specific user requirements.

View

- A view, like a table, is a database object.
- However, views are not "real" tables.
- They are logical representations of existing tables or of another view.
- Views contain no data of their own.
- They function as a window through which data from tables can be viewed or changed.

View

- The tables on which a view is based are called "base" tables.
- The view is a query stored as a SELECT statement in the data dictionary.

```
CREATE VIEW view_employees
AS SELECT employee_id emp_id,first_name,
last_name, email
FROM employees
WHERE employee_id BETWEEN 100 and 124;
```

```
SELECT *
FROM view_employees;
```

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL
100	Steven	King	SKING
101	Neena	Kochhar	NKOCHHAR
102	Lex	De Haan	LDEHAAN
103	Alexander	Hunold	AHUNOLD
104	Bruce	Ernst	BERNST
107	Diana	Lorentz	DLORENTZ
124	Kevin	Mourgos	KMOURGOS

Why Use Views?

- Views restrict access to base table data because the view can display selective columns from the table.
- Views can be used to reduce the complexity of executing queries based on more complicated SELECT statements.
- For example, the creator of the view can construct join statements that retrieve data from multiple tables.
- The user of the view neither sees the underlying code nor how to create it.
- The user, through the view, interacts with the database using simple queries.

Why Use Views?

- Views can be used to retrieve data from several tables, providing data independence for users.
- Users can view the same data in different ways.
- Views provide groups of users with access to data according to their particular permissions or criteria.

Creating a View

- To create a view, embed a subquery within the CREATE VIEW statement.
- The syntax of a view statement is as follows:

```
CREATE [OR REPLACE] [FORCE| NOFORCE] VIEW view  [(alias [,
   alias]...)] AS subquery
[WITH CHECK OPTION [CONSTRAINT constraint]]
[WITH READ ONLY [CONSTRAINT constraint]];
```

Creating a View

OR REPLACE	Re-creates the view if it already exists.
FORCE	Creates the view whether or not the base tables exist.
NOFORCE	Creates the view only if the base table exists (default).
view_name	Specifies the name of the view.
alias	Specifies a name for each expression selected by the view's query.
subquery	Is a complete SELECT statement. You can use aliases for the columns in the SELECT list. The subquery can contain complex SELECT syntax.

Creating a View

WITH CHECK OPTION	Specifies that rows remain accessible to the view after insert or update operations.
CONSTRAINT	Is the name assigned to the CHECK OPTION constraint.
WITH READ ONLY	Ensures that no DML operations can be performed on this view.

Creating a View

- Example:

```
CREATE OR REPLACE VIEW view_euro_countries
AS SELECT country_id, region_id, country_name, capitol
   FROM wf_countries
   WHERE location LIKE '%Europe';
```

```
SELECT * FROM view_euro_countries
ORDER BY country_name;
```

COUNTRY_ID	REGION_ID	COUNTRY_NAME	CAPITOL
22	155	Bailiwick of Guernsey	Saint Peter Port
203	155	Bailiwick of Jersey	Saint Helier
387	39	Bosnia and Herzegovina	Sarajevo
420	151	Czech Republic	Prague
298	154	Faroe Islands	Torshavn
49	155	Federal Republic of Germany	Berlin
33	155	French Republic	Paris
…	…	…	…

Guidelines for Creating a View

- The subquery that defines the view can contain complex SELECT syntax.
- The subquery that defines the view should not contain an ORDER BY clause. The ORDER BY clause is specified when you retrieve data from the view.
- You can use the OR REPLACE option to change the definition of the view without having to drop it or re-grant object privileges previously granted on it.
- Aliases can be used for the column names in the subquery.

CREATE VIEW Features

- Two classifications of views are used: simple and complex.
- The table summarizes the features of each view.

Feature	Simple Views	Complex Views
Number of tables used to derive data	One	One or more
Can contain functions	No	Yes
Can contain groups of data	No	Yes
Can perform DML operations (INSERT, UPDATE, DELETE) through a view	Yes	Not always

Simple View

- The view shown below is an example of a simple view.
- The subquery derives data from only one table and it does not contain a join function or any group functions.
- Because it is a simple view, INSERT, UPDATE, DELETE, and MERGE operations affecting the base table could possibly be performed through the view.

```
CREATE OR REPLACE VIEW view_euro_countries
AS SELECT country_id, country_name, capitol
   FROM wf_countries
   WHERE location LIKE '%Europe';
```

Simple View

- Column names in the SELECT statement can have aliases as shown below.
- Note that aliases can also be listed after the CREATE VIEW statement and before the SELECT subquery.

```
CREATE OR REPLACE VIEW view_euro_countries
AS SELECT country_id AS "ID", country_name AS "Country",
      capitol AS "Capitol City"
   FROM wf_countries
   WHERE location LIKE '%Europe';
```

```
CREATE OR REPLACE VIEW view_euro_countries("ID", "Country",
    "Capitol City")
AS SELECT country_id, country_name, capitol
   FROM wf_countries
   WHERE location LIKE '%Europe';
```

Simple View

- It is possible to create a view whether or not the base tables exist.
- Adding the word FORCE to the CREATE VIEW statement creates the view.
- As a DBA, this option could be useful during the development of a database, especially if you are waiting for the necessary privileges to the referenced object to be granted shortly.
- The FORCE option will create the view despite it being invalid.
- The NOFORCE option is the default when creating a view.

Complex View

- Complex views are views that can contain group functions and joins.
- The following example creates a view that derives data from two tables.

```
CREATE OR REPLACE VIEW view_euro_countries
   ("ID", "Country", "Capitol City", "Region")
AS SELECT c.country_id, c.country_name, c.capitol, r.region_name
   FROM wf_countries c JOIN wf_world_regions r
   USING (region_id)
   WHERE location LIKE '%Europe';
```

```
SELECT *
FROM view_euro_countries;
```

Complex View

ID	Country	Capitol City	Region
355	Republic of Albania	Tirana	Southern Europe
385	Republic of Croatia	Zagreb	Southern Europe
359	Republic of Bulgaria	Sofia	Southern Europe
378	Republic of San Marino	San Marino	Southern Europe
34	Kingdom of Spain	Madrid	Southern Europe
38	The Holy See (State of the Vatican City)	Vatican City	Southern Europe
30	Hellenic Republic	Athens	Southern Europe
386	Republic of Slovenia	Ljubljana	Southern Europe
381	Republic of Montenegro	Cetinje	Southern Europe
351	Portuguese Republic	Lisbon	Southern Europe
376	Principality of Andorra	Andorra la Vella	Southern Europe
350	Gibraltar	Gibraltar	Southern Europe
387	Bosnia and Herzegovina	Sarajevo	Southern Europe
…	…	…	…

Complex View

- Group functions can also be added to complex-view statements.

```
CREATE OR REPLACE VIEW view_high_pop
   ("Region ID", "Highest population")
AS SELECT region_id, MAX(population)
   FROM wf_countries
   GROUP BY region_id;
```

```
SELECT * FROM view_high_pop;
```

Region ID	Highest population
34	1095351995
151	142893540
30	1313973713
13	107449525
11	131859731
29	11382820
155	82422299
21	298444215
14	74777981
…	…

Modifying a View

- To modify an existing view without having to drop then re-create it, use the OR REPLACE option in the CREATE VIEW statement.
- The old view is replaced by the new version.
- For example:

```
CREATE OR REPLACE VIEW view_euro_countries
AS SELECT country_id, region_id, country_name, capitol
   FROM wf_countries
   WHERE location LIKE '%Europe';
```

Terminology

Key terms used in this lesson included:

- Alias
- Complex review
- CREATE VIEW
- FORCE
- NOFORCE
- REPLACE

Terminology

Key terms used in this lesson included:

- Simple view
- Subquery
- View
- VIEW_NAME

Summary

In this lesson, you should have learned how to:

- List three uses for views from the standpoint of a database administrator
- Explain, from a business perspective, why it is important to be able to create and use logical subsets of data derived from one or more tables
- Create a view with and without column aliases in the subquery using a single base table

Summary

In this lesson, you should have learned how to:

- Create a complex view that contains group functions to display values from two tables
- Retrieve data from a view


ORACLE
ACADEMY