

Database Programming with SQL 15-1: Creating Views

Practice Activities

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

# Vocabulary

Identify the vocabulary word for each definition below.

|  |  |
| --- | --- |
| **VIEW** | A subset of data from one or more tables that is generated from a query and stored as a virtual table |
| **VIEW\_NAME** | Name of view |
| **FORCE** | Creates a view regardless of whether or not the base tables exist |
| **SIMPLE VIEW** | Derives data from a table, no functions or groups, performs DML operations through the view |
| **NOFORCE** | Creates the view only if the base table exists |
| **CREATE VIEW statement** | Statement used to create a new view |
| **Alias** | Specifies a name for each expression selected by the view’s query |
| **subquery** | A complete SELECT statement |
| **Complex View** | Derives data from more than one table, contains functions or groups of data, and does not always allow DML operations through the view |
| **REPLACE** | Re-creates the view if it already exists |

# Try It / Solve It

1. What are three uses for a view from a DBA’s perspective?

**Restrict access and display selective columns**

·         **Reduce complexity of queries from other internal systems. So, providing a way to view same data in a different manner.**

·         **Let the app code rely on views and allow the internal implementation of tables to be modified later.**

1. Create a simple view called view\_d\_songs that contains the ID, title, and artist from the DJs on Demand table for each “New Age” type code. In the subquery, use the alias “Song Title” for the title column.

The subquery:

SELECT d\_songs.id, d\_songs.title "Song Title", d\_songs.artist

from d\_songs INNER JOIN d\_types ON d\_songs.type\_code = d\_types.code

where d\_types.description = 'New Age';

Now the view:

**CREATE VIEW view\_d\_songs AS**

**SELECT d\_songs.id, d\_songs.title "Song Title", d\_songs.artist**

**from d\_songs INNER JOIN d\_types ON d\_songs.type\_code = d\_types.code**

**where d\_types.description = 'New Age';**

But second execution of this SQL statement will say:

ORA-00955: name is already used by an existing object

If I want to recreate the view/ create it first time

CREATE OR REPLACE VIEW view\_d\_songs AS

SELECT d\_songs.id, d\_songs.title "Song Title", d\_songs.artist

from d\_songs INNER JOIN d\_types ON d\_songs.type\_code = d\_types.code

where d\_types.description = 'New Age';

Verify results again:

SELECT \* FROM view\_d\_songs ;

1. SELECT \*

FROM view\_d\_songs. What was returned?

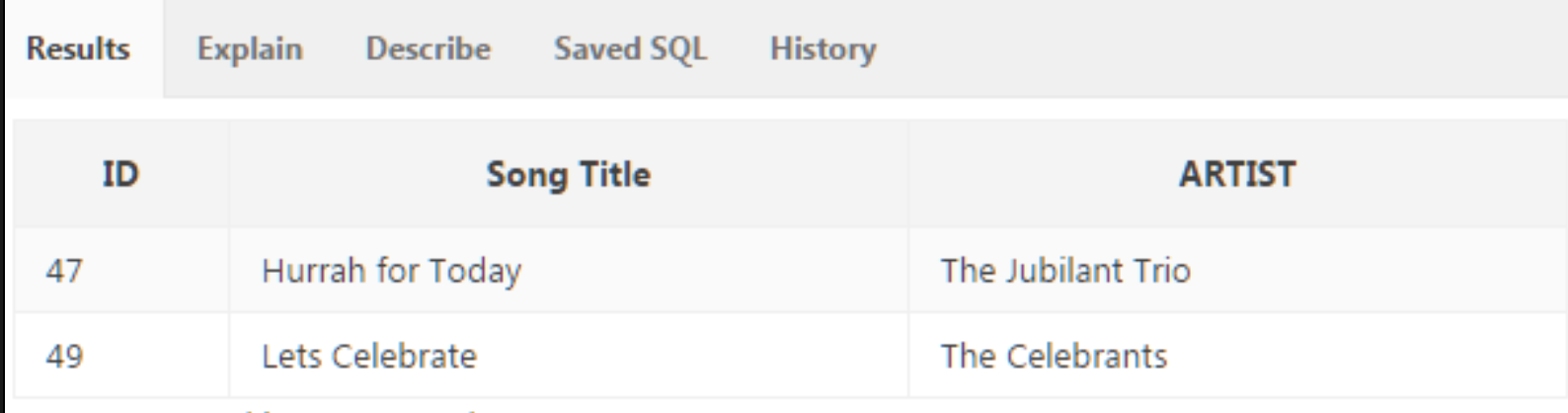
The result is same as that of

SELECT d\_songs.id, d\_songs.title "Song Title", d\_songs.artist

from d\_songs INNER JOIN d\_types ON d\_songs.type\_code = d\_types.code

where d\_types.description = 'New Age';

**SELECT \* FROM view\_d\_songs ;**



1. REPLACE view\_d\_songs. Add type\_code to the column list. Use aliases for all columns.

**CREATE OR REPLACE VIEW view\_d\_songs AS**

**SELECT d\_songs.id, d\_songs.title "Song Title", d\_songs.artist, d\_songs.type\_code**

**from d\_songs INNER JOIN d\_types ON d\_songs.type\_code = d\_types.code**

**where d\_types.description = 'New Age';**

1. Jason Tsang, the disk jockey for DJs on Demand, needs a list of the past events and those planned for the coming months so he can make arrangements for each event’s equipment setup. As the company manager, you do not want him to have access to the price that clients paid for their events. Create a view for Jason to use that displays the name of the event, the event date, and the theme description. Use aliases for each column name.

**CREATE OR REPLACE VIEW view\_d\_events\_pkgs AS**

**SELECT evt.name "Name of Event", TO\_CHAR(evt.event\_date, 'dd-Month-yyyy')  "Event date", thm.description "Theme description"**

**FROM  d\_events  evt INNER JOIN d\_themes  thm  ON evt.theme\_code = thm.code**

**WHERE evt.event\_date <= ADD\_MONTHS(SYSDATE,1);**

SELECT \* FROM view\_d\_events\_pkgs ;

1. It is company policy that only upper-level management be allowed access to individual employee salaries. The department managers, however, need to know the minimum, maximum, and average salaries, grouped by department. Use the Oracle database to prepare a view that displays the needed information for department managers.

DESCRIBE employees;

suggests:

Salary is a nullable field, I don't want to miss nulls in average/min/max calculation.

There may be some employees without department mentioned since it is nullable. I want to miss such records in my calculations.

SELECT department\_id FROM departments WHERE department\_id NOT IN ( SELECT NVL(department\_id,0) FROM employees);

Suggests:

There may be a department for which there is no record in employees table.

**CREATE OR REPLACE VIEW view\_min\_max\_avg\_dpt\_salary ("Department Id", "Department Name", "Max Salary", "Min Salary", "Average Salary") AS**

**SELECT dpt.department\_id, dpt.department\_name, MAX(NVL(emp.salary,0)), MIN(NVL(emp.salary,0)), ROUND(AVG(NVL(emp.salary,0)),2)**

**FROM departments dpt LEFT OUTER JOIN employees emp ON dpt.department\_id = emp.department\_id**

**GROUP BY (dpt.department\_id, dpt.department\_name);**

SELECT \* FROM view\_min\_max\_avg\_dpt\_salary;

Copyright © 2020, Oracle and/or its affiliates. All rights reserved. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.