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.

2. 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.

```
SELECT * FROM view_d_songs ;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';
```

3. SELECT *
   FROM view_d_songs.

   What was returned?

```
SELECT * FROM view_d_songs ;
```

4.

```
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';
```

4. REPLACE view_d_songs. Add type_code to the column list. Use aliases for all columns.

5. 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.

6. 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.

5.

```
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 ;
```