Creating Views
Objectives

After completing this lesson, you should be able to do the following:

- Describe a view
- Create, alter the definition of, and drop a view
- Retrieve data through a view
- Insert, update, and delete data through a view
- Create and use an inline view
- Perform “Top-N” analysis
# Database Objects

<table>
<thead>
<tr>
<th>Object</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td>Basic unit of storage; composed of rows and columns</td>
</tr>
<tr>
<td>View</td>
<td>Logically represents subsets of data from one or more tables</td>
</tr>
<tr>
<td>Sequence</td>
<td>Generates primary key values</td>
</tr>
<tr>
<td>Index</td>
<td>Improves the performance of some queries</td>
</tr>
<tr>
<td>Synonym</td>
<td>Alternative name for an object</td>
</tr>
</tbody>
</table>
What is a View?

**EMPLOYEES Table:**

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>FIRST_NAME</th>
<th>LAST_NAME</th>
<th>EMAIL</th>
<th>PHONE_NUMBER</th>
<th>HIRE_DATE</th>
<th>JOB_ID</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>Steven</td>
<td>King</td>
<td>SKING</td>
<td>515.123.4567</td>
<td>17-JUN-87</td>
<td>AD_PRES</td>
<td>240</td>
</tr>
<tr>
<td>101</td>
<td>Neena</td>
<td>Kochhar</td>
<td>NKOCCHAR</td>
<td>515.123.4568</td>
<td>21-SEP-89</td>
<td>AD_VP</td>
<td>170</td>
</tr>
<tr>
<td>102</td>
<td>Lex</td>
<td>De Haan</td>
<td>LDEHAAN</td>
<td>515.123.4569</td>
<td>13-JAN-93</td>
<td>AD_VP</td>
<td>170</td>
</tr>
<tr>
<td>103</td>
<td>Alexander</td>
<td>Hunold</td>
<td>AHUNOLD</td>
<td>590.423.4567</td>
<td>03-JAN-90</td>
<td>IT_PROG</td>
<td>90</td>
</tr>
<tr>
<td>104</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>60</td>
</tr>
<tr>
<td>105</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>42</td>
</tr>
<tr>
<td>106</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>58</td>
</tr>
<tr>
<td>107</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>35</td>
</tr>
<tr>
<td>108</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>45</td>
</tr>
<tr>
<td>109</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>110</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
</tr>
<tr>
<td>111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>Zlotkey</td>
<td>10500</td>
</tr>
<tr>
<td>174</td>
<td>Abel</td>
<td>11000</td>
</tr>
<tr>
<td>176</td>
<td>Taylor</td>
<td>6600</td>
</tr>
<tr>
<td>170</td>
<td>Kirkby</td>
<td></td>
</tr>
<tr>
<td>200</td>
<td>Jennifer</td>
<td></td>
</tr>
<tr>
<td>201</td>
<td>Michael</td>
<td></td>
</tr>
<tr>
<td>202</td>
<td>Pat</td>
<td></td>
</tr>
<tr>
<td>205</td>
<td>Shelley</td>
<td></td>
</tr>
<tr>
<td>206</td>
<td>William</td>
<td></td>
</tr>
</tbody>
</table>

20 rows selected.
Why Use Views?

- To restrict data access
- To make complex queries easy
- To provide data independence
- To present different views of the same data
## Simple Views and Complex Views

<table>
<thead>
<tr>
<th>Feature</th>
<th>Simple Views</th>
<th>Complex Views</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of tables</td>
<td>One</td>
<td>One or more</td>
</tr>
<tr>
<td>Contain functions</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Contain groups of data</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>DML operations through a view</td>
<td>Yes</td>
<td>Not always</td>
</tr>
</tbody>
</table>
Creating a View

- You embed a subquery within the CREATE VIEW statement.

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

- The subquery can contain complex SELECT syntax.
Creating a View

- Create a view, EMPVU80, that contains details of employees in department 80.

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
```

View created.

- Describe the structure of the view by using the iSQL*Plus DESCRIBE command.

```
DESCRIBE empvu80
```
Creating a View

- Create a view by using column aliases in the subquery.

```
CREATE VIEW salvu50
AS SELECT employee_id ID_NUMBER, last_name NAME, 
        salary*12 ANN_SALARY
        FROM employees
        WHERE department_id = 50;
```

View created.

- Select the columns from this view by the given alias names.
Retrieving Data from a View

```
SELECT *
FROM salvu50;
```
Querying a View

**Oracle Server**

**USER_VIEWS**

```
EMPVU80
SELECT employee_id, last_name, salary
FROM employees
WHERE department_id=80;
```

---

**iSQL*Plus**

```
SELECT * from empvu80;
```

---

<table>
<thead>
<tr>
<th>EMPLOYEE_ID</th>
<th>LAST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>149</td>
<td>Zlotkey</td>
<td>10500</td>
</tr>
<tr>
<td>174</td>
<td>Abel</td>
<td>11000</td>
</tr>
<tr>
<td>176</td>
<td>Taylor</td>
<td>6600</td>
</tr>
</tbody>
</table>
Modifying a View

• Modify the EMPVU80 view by using CREATE OR REPLACE VIEW clause. Add an alias for each column name.

CREATE OR REPLACE VIEW empvuu80
(id_number, name, sal, department_id)
AS SELECT employee_id, first_name || ' ' || last_name,
       salary, department_id
FROM employees
WHERE department_id = 80;

View created.

• Column aliases in the CREATE VIEW clause are listed in the same order as the columns in the subquery.
Creating a Complex View

Create a complex view that contains group functions to display values from two tables.

```
CREATE VIEW dept_sum_vu
  (name, minsal, maxsal, avgsal)
AS SELECT d.department_name, MIN(e.salary),
         MAX(e.salary), AVG(e.salary)
FROM employees e, departments d
WHERE e.department_id = d.department_id
GROUP BY d.department_name;
```

View created.
Rules for Performing DML Operations on a View

- You can perform DML operations on simple views.
- You cannot remove a row if the view contains the following:
  - Group functions
  - A GROUP BY clause
  - The DISTINCT keyword
  - The pseudocolumn ROWNUM keyword
Rules for Performing DML Operations on a View

You cannot modify data in a view if it contains:

• Group functions
• A GROUP BY clause
• The DISTINCT keyword
• The pseudocolumn ROWNUM keyword
• Columns defined by expressions
Rules for Performing DML Operations on a View

You cannot add data through a view if the view includes:

- Group functions
- A `GROUP BY` clause
- The `DISTINCT` keyword
- The pseudocolumn `ROWNUM` keyword
- Columns defined by expressions
- `NOT NULL` columns in the base tables that are not selected by the view
Using the WITH CHECK OPTION Clause

- You can ensure that DML operations performed on the view stay within the domain of the view by using the WITH CHECK OPTION clause.

```sql
CREATE OR REPLACE VIEW empvu20
AS SELECT *
    FROM employees
    WHERE department_id = 20
    WITH CHECK OPTION CONSTRAINT empvu20_ck;
```

View created.

- Any attempt to change the department number for any row in the view fails because it violates the WITH CHECK OPTION constraint.
Denying DML Operations

• You can ensure that no DML operations occur by adding the **WITH READ ONLY** option to your view definition.

• Any attempt to perform a DML on any row in the view results in an Oracle server error.
Denying DML Operations

CREATE OR REPLACE VIEW empvu10
  (employee_number, employee_name, job_title)
AS SELECT employee_id, last_name, job_id
  FROM employees
WHERE department_id = 10
WITH READ ONLY;

View created.
Removing a View

You can remove a view without losing data because a view is based on underlying tables in the database.

DROP VIEW view;

DROP VIEW empvu80; 
View dropped.
Inline Views

• An inline view is a subquery with an alias (or correlation name) that you can use within a SQL statement.

• A named subquery in the FROM clause of the main query is an example of an inline view.

• An inline view is not a schema object.
Top-N Analysis

- Top-N queries ask for the $n$ largest or smallest values of a column. For example:
  - What are the ten best selling products?
  - What are the ten worst selling products?
- Both largest values and smallest values sets are considered Top-N queries.
Performing Top-N Analysis

The high-level structure of a Top-N analysis query is:

```
SELECT [column_list], ROWNUM
FROM (SELECT [column_list]
       FROM table
       ORDER BY Top-N_column)
WHERE ROWNUM <= N;
```
Example of Top-N Analysis

To display the top three earner names and salaries from the EMPLOYEES table:

```
SELECT ROWNUM as RANK, first_name, salary
FROM (SELECT last_name, salary FROM employees
      ORDER BY salary DESC)
WHERE ROWNUM <= 3;
```

<table>
<thead>
<tr>
<th>RANK</th>
<th>LAST_NAME</th>
<th>SALARY</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>King</td>
<td>24000</td>
</tr>
<tr>
<td>2</td>
<td>Kochhar</td>
<td>17000</td>
</tr>
<tr>
<td>3</td>
<td>De Haan</td>
<td>17000</td>
</tr>
</tbody>
</table>
Summary

In this lesson, you should have learned that a view is derived from data in other tables or views and provides the following advantages:

- Restricts database access
- Simplifies queries
- Provides data independence
- Provides multiple views of the same data
- Can be dropped without removing the underlying data
- An inline view is a subquery with an alias name.
- Top-N analysis can be done using subqueries and outer queries.
Practice 11 Overview

This practice covers the following topics:

- Creating a simple view
- Creating a complex view
- Creating a view with a check constraint
- Attempting to modify data in the view
- Displaying view definitions
- Removing views