

Creating Views

ORACLE[®]

Copyright © 2019, Oracle and/or its affiliates. All rights reserved.

Objectives

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

- Create simple and complex views
- Retrieve data from views
- Querying the dictionary views for the view information

In this lesson, you are introduced to views, and you learn the basics of creating and using views.

Lesson Agenda

- Overview of views
- Creating, modifying, and retrieving data from a view
- Querying the dictionary views for view information
- Data manipulation language (DML) operations on a view
- Dropping a view

Database Objects

| Object | Description |
|----------|--|
| Table | Basic unit of storage; composed of rows |
| View | Logically represents subsets of data from one or more tables |
| Sequence | Generates numeric values |
| Index | Improves the performance of data retrieval queries |
| Synonym | Gives alternative names to objects |

ORACLE

There are several other objects in a database in addition to tables.

With views, you can present and hide data from the tables.

Many applications require the use of unique numbers as primary key values. You can either build code into the application to handle this requirement or use a sequence to generate unique numbers.

If you want to improve the performance of data retrieval queries, you should consider creating an index. You can also use indexes to enforce uniqueness on a column or a collection of columns.

You can provide alternative names for objects by using synonyms.

What Is a View?

- You can present logical subsets or combinations of data by creating views of tables.
- A view is a schema object , a stored `SELECT` statement based on a table or another view.
- A view contains no data of its own, but is like a window through which data from tables can be viewed or changed.
- The tables on which a view is based are called *base tables*. The view is stored as a `SELECT` statement in the data dictionary.

What Is a View?

EMPLOYEES table

| EMPLOYEE_ID | FIRST_NAME | LAST_NAME | EMAIL | PHONE_NUMBER | HIRE_DATE | JOB_ID | SALARY |
|-------------|------------|-----------|----------|--------------|-----------|---------|--------|
| 100 | Steven | King | SKING | 515.123.4567 | 17-JUN-87 | AD_PRES | 24000 |
| 101 | Neena | Kochhar | NKOCHHAR | 515.123.4568 | 21-SEP-89 | AD_VP | 17000 |
| 102 | Lex | De Haan | LDEHAAN | 515.123.4569 | 13-JAN-93 | AD_VP | 17000 |
| 103 | Alexander | Hunold | AHUNOLD | 590.423.4567 | 03-JAN-90 | IT_PROG | 9000 |
| 104 | Bruce | Ernst | BERNST | 590.423.5666 | 17-FEB-96 | IT_PROG | 6000 |
| 105 | David | Fay | DFAY | 590.423.5666 | 17-AUG-97 | IT_PROG | 4200 |
| 106 | Allen | King | AKING | 590.423.5666 | 17-SEP-97 | IT_PROG | 6900 |
| 107 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 5800 |
| 108 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 3500 |
| 109 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 3100 |
| 110 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 2600 |
| 111 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 2500 |
| 112 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 10500 |
| 113 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 11000 |
| 114 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 8600 |
| 115 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 7000 |
| 116 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 4400 |
| 117 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 13000 |
| 118 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 6000 |
| 119 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 12000 |
| 120 | Part | Simon | PSIMON | 590.423.5666 | 17-SEP-97 | IT_PROG | 8300 |

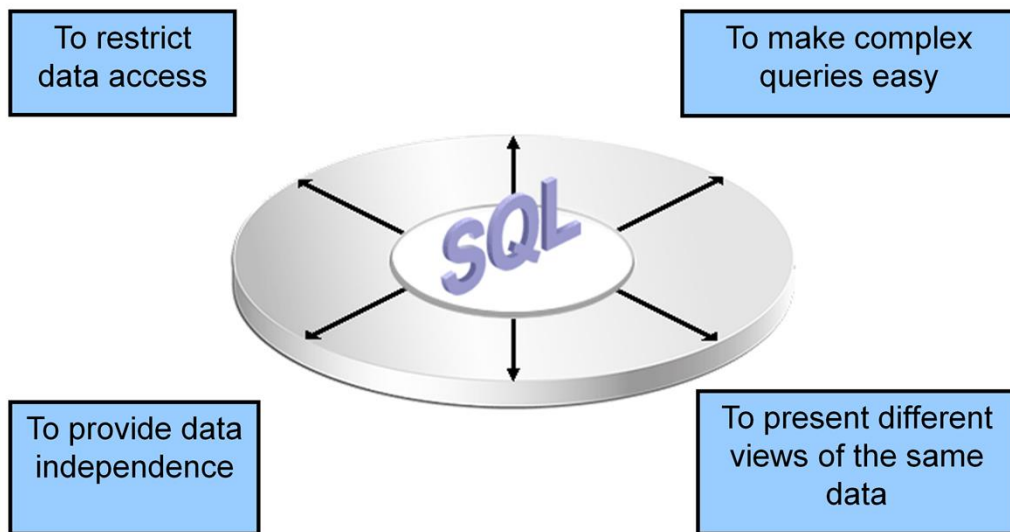
ORACLE

You can present logical subsets or combinations of data by creating views of tables. A view is a schema object, a stored `SELECT` statement based on a table or another view. A view contains no data of its own, but is like a window through which data from tables can be viewed or changed. The tables on which a view is based are called *base tables*. The view is stored as a `SELECT` statement in the data dictionary.

Advantages of Views

- Views restrict access to the data because they display selected columns from the table.
- Views can be used to make simple queries to retrieve the results of complicated queries. For example, views can be used to query information from multiple tables without the user knowing how to write a join statement.
- Views provide data independence for ad hoc users and application programs. One view can be used to retrieve data from several tables.
- Views provide groups of users access to data according to their particular criteria.

Advantages of Views



ORACLE

- Views restrict access to the data because they display selected columns from the table.
- Views can be used to make simple queries to retrieve the results of complicated queries. For example, views can be used to query information from multiple tables without the user knowing how to write a join statement.
- Views provide data independence for ad hoc users and application programs. One view can be used to retrieve data from several tables.
- Views provide groups of users access to data according to their particular criteria.

For more information, see the “`CREATE VIEW`” section in *Oracle Database SQL Language Reference* for Oracle Database 19c.

Simple Views and Complex Views

| Feature | Simple Views | Complex Views |
|-------------------------------|--------------|---------------|
| Number of tables | One | One or more |
| Contain functions | No | Yes |
| Contain groups of data | No | Yes |
| DML operations through a view | Yes | Not always |

ORACLE

There are two classifications for views: simple and complex. The basic difference is related to the DML (`INSERT`, `UPDATE`, and `DELETE`) operations.

- A simple view is one that:
 - Derives data from only one table
 - Contains no functions or groups of data
 - Can perform DML operations through the view
- A complex view is one that:
 - Derives data from many tables
 - Contains functions or groups of data
 - Does not always allow DML operations through the view

Creating a View

- You embed a subquery in 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.

ORACLE

You can create a view by embedding a subquery in the `CREATE VIEW` statement.

In the syntax:

| | |
|--------------------------------|--|
| <code>OR REPLACE</code> | Re-creates the view if it already exists. You can use this clause to change the definition of an existing view without dropping, re-creating, and regranteeing object privileges previously granted on it. |
| <code>FORCE</code> | Creates the view regardless of whether or not the base tables exist |
| <code>NOFORCE</code> | Creates the view only if the base tables exist (This is the default.) |
| <code>view</code> | Is the name of the view |
| <code>alias</code> | Specifies names for the expressions selected by the view's query (The number of aliases must match the number of expressions selected by the view.) |
| <code>subquery</code> | Is a complete <code>SELECT</code> statement (You can use aliases for the columns in the <code>SELECT</code> list.) |
| <code>WITH CHECK OPTION</code> | Specifies that only those rows that are accessible to the view can be inserted or updated |
| <code>Constraint</code> | Is the name assigned to the <code>CHECK OPTION</code> constraint |
| <code>WITH READ ONLY</code> | Ensures that no DML operations can be performed on this view |

Note: In SQL Developer, click the Run Script icon or press F5 to run the data definition language (DDL) statements. The feedback messages will be shown on the Script Output tabbed page.

- **REPLACE** You can use this clause to change the definition of an existing view without dropping, re-creating, and re-granting object privileges previously granted on it.
- **FORCE** Creates the view regardless of whether or not the base tables exist
- **NOFORCE** Creates the view only if the base tables exist (This is the default.)
- **WITH CHECK OPTION** Specifies that only those rows that are accessible to the view can be inserted or updated
- *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

- Create the EMPVU80 view, which contains details of the employees in department 80:

```
CREATE VIEW empvu80
AS SELECT employee_id, last_name, salary
FROM employees
WHERE department_id = 80;
view EMPVU80 created.
```

- Describe the structure of the view by using the SQL*Plus DESCRIBE command:

```
DESCRIBE empvu80;
```

The example in the slide creates a view that contains the employee number, last name, and salary for each employee in department 80.

You can display the structure of the view by using the DESCRIBE command.

```
DESCRIBE empvu80
Name      Null    Type
-----
EMPLOYEE_ID NOT NULL  NUMBER(6)
LAST_NAME   NOT NULL  VARCHAR2(25)
SALARY                        NUMBER(8,2)
```

Guidelines

- The subquery that defines a view can contain complex SELECT syntax, including joins, groups, and subqueries.
- If you do not specify a constraint name for the view created with the WITH CHECK OPTION, the system assigns a default name in the SYS_Cn format.
- You can use the OR REPLACE option to change the definition of the view without dropping and re-creating it, or regranteeing the object privileges previously granted on it.

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 SALVU50 created.

- Select the columns from this view by the given alias names.

You can control the column names by including column aliases in the subquery.

The example in the slide creates a view containing the employee number (`EMPLOYEE_ID`) with the alias `ID_NUMBER`, name (`LAST_NAME`) with the alias `NAME`, and annual salary (`SALARY`) with the alias `ANN_SALARY` for every employee in department 50.

Alternatively, you can use an alias after the `CREATE` statement and before the `SELECT` subquery. The number of aliases listed must match the number of expressions selected in the subquery.

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

view SALVU50 created.

Retrieving Data from a View

```
SELECT *  
FROM salvu50;
```

| | ID_NUMBER | NAME | ANN_SALARY |
|---|-----------|-------------|------------|
| 1 | 120 | Weiss | 96000 |
| 2 | 121 | Fripp | 98400 |
| 3 | 122 | Kaufling | 94800 |
| 4 | 123 | Vollman | 78000 |
| 5 | 124 | Mourgos | 69600 |
| 6 | 125 | Nayer | 38400 |
| 7 | 126 | Mikkilineni | 32400 |

...

ORACLE

You can retrieve data from a view as you would from any table. You can display either the contents of the entire view or just specific rows and columns.

Modifying a View

- **Modify the EMPVU80 view by using a CREATE OR REPLACE VIEW clause. Add an alias for each column name:**

```
CREATE OR REPLACE VIEW empvu80
(id_number, name, sal, department_id)
AS SELECT  employee_id, first_name || ' '
           || last_name, salary, department_id
FROM      employees
WHERE     department_id = 80;
```

view EMPVU80 created.

- **Column aliases in the CREATE OR REPLACE VIEW clause are listed in the same order as the columns in the subquery.**

ORACLE

With the `OR REPLACE` option, a view can be created even if one exists with this name already, thus replacing the old version of the view for its owner. This means that the view can be altered without dropping, re-creating, and regranting object privileges.

Note: When assigning column aliases in the `CREATE OR REPLACE VIEW` clause, remember that the aliases 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 OR REPLACE 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 JOIN departments d
ON            (e.department_id = d.department_id)
GROUP BY d.department name;
```

```
view DEPT_SUM_VU created.
```

The example in the slide creates a complex view of department names, minimum salaries, maximum salaries, and the average salaries by department. Note that alternative names have been specified for the view. This is a requirement if any column of the view is derived from a function or an expression.

You can view the structure of the view by using the `DESCRIBE` command. Display the contents of the view by issuing a `SELECT` statement.

```
SELECT  *
FROM    dept sum vu;
```

| A | B | C | D |
|----|------------------|--------|--------|
| | NAME | MINSAL | MAXSAL |
| 1 | Administration | 4400 | 4400 |
| 2 | Accounting | 8300 | 12008 |
| 3 | Purchasing | 2500 | 11000 |
| 4 | Human Resources | 6500 | 6500 |
| 5 | IT | 4200 | 9000 |
| 6 | Public Relations | 10000 | 10000 |
| 7 | Executive | 17000 | 24000 |
| 8 | Shipping | 2100 | 8200 |
| 9 | Sales | 6100 | 14000 |
| 10 | Finance | 6900 | 12008 |
| 11 | Marketing | 6000 | 13000 |

Data dictionary view (USER_VIEWS)

1

```
DESCRIBE user_views
```

| Name | Null | Type |
|-------------|----------|--------------|
| VIEW_NAME | NOT NULL | VARCHAR2(30) |
| TEXT_LENGTH | | NUMBER |
| TEXT | | LONG() |

...

2

```
SELECT view_name FROM user_views;
```

| VIEW_NAME |
|--------------------|
| 1 EMP_DETAILS_VIEW |
| 2 SALVU50 |
| 3 EMPVU80 |
| 4 DEPT_SUM_VU |

3

```
SELECT text FROM user_views  
WHERE view_name = 'EMP_DETAILS_VIEW';
```

| TEXT |
|--|
| 1 SELECT e.employee_id, e.job_id, e.manager_id, e.department_id, d.location_id, l.co |

...

```
AND c.region_id = r.region_id AND j.job_id = e.job_id WITH READ ONLY
```

ORACLE

After your view is created, you can query the data dictionary view called `USER_VIEWS` to see the name of the view and the view definition. The text of the `SELECT` statement that constitutes your view is stored in a `LONG` column. The `LENGTH` column is the number of characters in the `SELECT` statement. By default, when you select from a `LONG` column, only the first 80 characters of the column's value are displayed. To see more than 80 characters in SQL*Plus, use the `SET LONG` command:

```
SET LONG 1000
```

In the examples in the slide:

1. The `USER_VIEWS` columns are displayed. Note that this is a partial listing.
2. The names of your views are retrieved
3. The `SELECT` statement for the `EMP_DETAILS_VIEW` is displayed from the dictionary



Data Access Using Views

When you access data by using a view, the Oracle Server performs the following operations:

- It retrieves the view definition from the data dictionary table `USER_VIEWS`.
- It checks access privileges for the view base table.
- It converts the view query into an equivalent operation on the underlying base table or

tables. That is, data is retrieved from, or an update is made to, the base tables.

Rules for Performing DML Operations on a View

- You can usually 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

- You can perform DML operations on data through a view if those operations follow certain rules.
- You can remove a row from a view unless it contains any of 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
 - (for example, `SALARY * 12`).

ORACLE

You can modify data through a view unless it contains any of the conditions mentioned in the previous slide or columns defined by expressions (for example, `SALARY * 12`).

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 without default value in the base tables that are not selected by the view

ORACLE

You can add data through a view unless it contains any of the items listed in the slide. You cannot add data to a view if the view contains `NOT NULL` columns without default values in the base table. All the required values must be present in the view. Remember that you are adding values directly to the underlying table *through* the view.

For more information, see the “`CREATE VIEW`” section in *Oracle Database SQL Language Reference* for Oracle Database 19c.

Using the WITH CHECK OPTION Clause

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

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

view EMPVU20 created.

- Any attempt to `INSERT` a row with a `department_id` other than 20 or to `UPDATE` the department number for any row in the view fails because it violates the `WITH CHECK OPTION` constraint.

It is possible to perform referential integrity checks through views. You can also enforce constraints at the database level. The view can be used to protect data integrity, but the use is very limited.

The `WITH CHECK OPTION` clause specifies that `INSERTs` and `UPDATES` performed through the view cannot create rows that the view cannot select. Therefore, it enables integrity constraints and data validation checks to be enforced on data being inserted or updated. If there is an attempt to perform DML operations on rows that the view has not selected, an error is displayed, along with the constraint name if that has been specified.

```
UPDATE empvu20
SET     department_id = 10
WHERE   employee_id = 201;
```

Error:

```
SQL Error: ORA-01402: view WITH CHECK OPTION where-clause violation
01402. 00000 - "view WITH CHECK OPTION where-clause violation"
*Cause:
*Action:
```

Note: No rows are updated because, if the department number were to change to 10, the

view would no longer be able to see that employee. With the `WITH CHECK OPTION` clause, therefore, the view can see only the employees in department 20 and does not allow the department number for those employees to be changed through the view.

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 operation on any row in the view results in an Oracle server error.



ORACLE

You can ensure that no DML operations occur on your view by creating it with the `WITH READ ONLY` option. The example in the next slide modifies the `EMPVU10` view to prevent any DML operations on the view.

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 EMPVU10 created.
```

- Any attempt to remove/insert/update a row from a view with a read-only constraint results in an error:

Any attempt to remove a row from a view with a read-only constraint results in an error:

```
DELETE FROM empvu10
WHERE employee_number = 200;
```

Similarly, any attempt to insert a row or modify a row using the view with a read-only constraint results in the same error.

```
Error report:
SQL Error: ORA-42399: cannot perform a DML operation on a read-only view
```

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 EMPVU80 dropped.
```

Only the creator or a user with the `DROP ANY VIEW` privilege can remove a view.

ORACLE

You use the `DROP VIEW` statement to remove a view. The statement removes the view definition from the database. However, dropping views has no effect on the tables on which the view was based. Alternatively, views or other applications based on the deleted views become invalid. Only the creator or a user with the `DROP ANY VIEW` privilege can remove a view.

In the syntax, *view* is the name of the view.

Quiz

You cannot add data through a view if the view includes the pseudocolumn `ROWNUM` keyword

- a. True
- b. False

Answer: a

Summary

In this lesson, you should have learned how to:

- Create, use, and remove views
- Querying the dictionary views for view information

In this lesson, you should have learned about views.