Chapter 10
VIEWS

- ❑ What is a view?
- ❑ Why we need a view?
- ❑ Creating and using a view
- ❑ Simplifying query using view
- ❑ Presenting data in different forms
- ❑ Isolating application from changes in definition of table
- ❑ Changing base table through view
- ❑ Updatable join views
- ❑ Dropping a view

In this chapter, we will understand what is a view and how to create and use it. View is certainly one of the most important schema objects of Oracle database. It is important to understand what a view can do in Oracle.

What Is A View?
A view is a window through which you access a portion of one or more tables. View itself doesn't contain any data but it refers to the data of a table on which it is based.

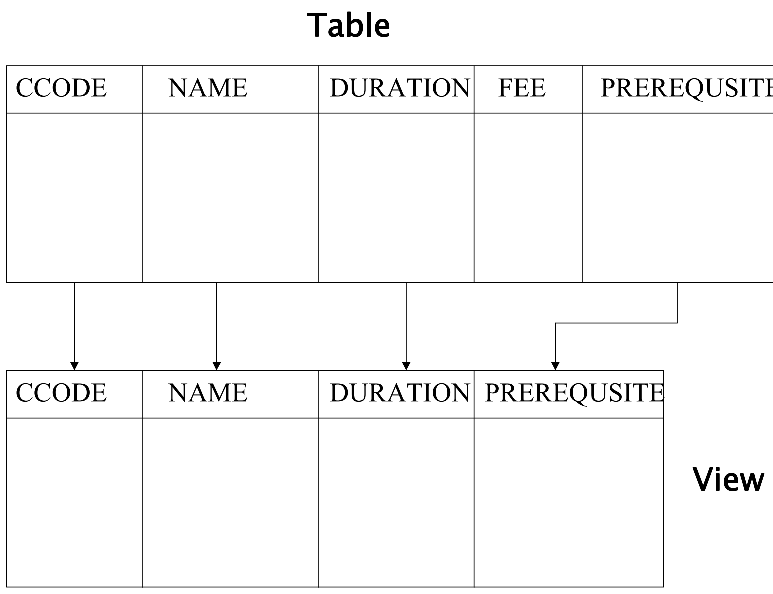

Figure 1: Example of a view.

The table on which the view is based is called as **base table**.

A view is also called as "virtual table" as it can be used when a table can be used. View retrieves the data by executing the query and presents the data in the form of a table.

Views are used to let users access only the portion of the data that they are supposed to access. Views are very commonly used objects in Oracle. In fact objects such as USER_TABLES, USER_CONSTRAINTS etc., are not actually tables and instead they are views.

Why We Need A View?

A view is required in several cases. Let us start with a simple requirement. Assume the table COURSES is owned by user "FIGO". As the owner of the table Figo will have all the privileges on that table. User Figo now wants another user – Raul to access COURSES table. To this effect Figo can grant permission to Raul. But Figo doesn't want Raul to access column FEE.

If Figo grants permission to Raul then Raul will have complete access to table and Figo cannot stop Raul from accessing FEE column. So Figo creates a view on COURSES table that includes everything from COURSES table except FEE. Then Figo grants permission on the view to Raul and not on the base table. Since Raul has access only to the view, he can access whatever data is presented by the view. See figure 1.

A view in the above requirement is quite ideal for two reasons.

1. It fulfills the requirement without storing a separate copy of the data. A view doesn't store any data of its own and takes the data from base table.
2. As the data is taken from base table, accurate and up-to-date information is provided to Raul. Yet the column to be hidden from Raul is hidden as it is not part of the view.

The following are the other important applications of views:

- Provides an extra layer on the top of table allowing only a predetermined rows or columns to be accessed.
- Allows complex queries to be stored in the database. A view stores the query that is used to create it in the database. It uses the query to retrieve the data from the base table(s). If a complex query is to be referred again and again then it can be stored in the form of a view.

- Can present the data of the table in different forms. For instance, the name of the columns can be changed and two or more columns can be presented as one column or split one column as two or more columns.
- Can isolate application from the changes in the definition of the table.

In the next section let us see how to create and use view.

Creating and using views

A view is created using CREATE VIEW command. At the time of creating the view, we have to give the name of the view and the query on which the view is based.

For example, to create a view that takes everything from COURSES table except FEE column, given the following CREATE TABLE command.

```
SQL> create view course_view
  2  as
  3    select ccode,name,duration, prerequisite from courses;

View created.
```

In the above example, COURSE_VIEW is the name of the view and COURSES is the base table. COURSE_VIEW view can access all columns of the table except FEE column.

Here is the syntax of CREATE VIEW command.

```
CREATE  [OR REPLACE]  [FORCE ] VIEW  viewname
           [(column-name, column-name)]
           AS Query
           [with check option];
```

Note: *ORDER BY clause cannot be used in query used with CREATE VIEW command.*

OR REPLACE Option

Allows a view to be created even if one already exists. This allows altering view without dropping, recreating and re-granting object privileges.

FORCE Option

Allows view to be created even if the base table doesn't exist. However, the base table should exist before the view is used.

I will discuss about WITH CHECK OPTION later in this chapter.

Once a view is created, a view can be used similar to a table. For example, you can use SELECT command with COURSE_VIEW as follows:

```
select * from course_view;

CCODE NAME                            DURATION PREREQUISITE
----- ------------------------------ --------- -------------------
-----------
ora   Oracle database                       25 Windows
vbnet VB.NET                                30 Windows and
programming
c     C programming                         20 Computer Awareness
asp   ASP.NET                               25 Internet and
programming
java  Java Language                         25 C language
xml   XML Programming                       15 HTML,Scripting,
ASP/JSP
cs    C Sharp                               30 C Language
```

As we have seen before, a view can be used just like a table. That is the reason a view is called as **virtual table**.

It is also possible to select columns of a view, as it is illustrated below:

```
select  name, duration from course_view;

NAME                            DURATION
------------------------------ ---------
Oracle database                       25
VB.NET                                30
C programming                         20
ASP.NET                               25
Java Language                         25
XML Programming                       15
C Sharp                               30
```

In the same way it is possible to filter rows just like how it is done with tables.

```
select * from course_view
where  duration > 25;

CCODE NAME                            DURATION PREREQUISITE
----- ------------------------------ --------- -------------------
-----------
vbnet VB.NET                                30 Windows and
programming
cs    C Sharp                               30 C Language
```

Note: *Though I said, a view is same as a table, it is not always right. In this chapter and in later chapters, you will come to know where, when and how a view is different from a table.*

Simplifying query using view

A view apart from providing access control can also be used to simplify query. A view is also called as "stored query". The query given at the time of creating view is stored by Oracle and used whenever the view is used. All that you get in the view is the data retrieved by the query.

Since a query is stored for view, a complex query can be stored in the form of a view. The following example creates

```
create view coursecount
as
select ccode, count(*) nobatches , max(stdate)  lastdate
from batches
group by ccode;
```

Now instead of giving the lengthy query, it is possible to get the details using view COURSECOUNT as follows:

```
SQL> select * from coursecount;

CCODE NOBATCHES LASTDATE
----- --------- ---------
asp           1 15-JAN-01
c             1 20-JAN-01
java          1 05-APR-01
ora           2 15-AUG-01
vbnet         1 12-JUL-01
xml           1 02-MAR-01
```

If query used to create the view contains any expression then the expression must be given alias. The alias of the expression becomes the name of the column in the view.

In the above example the expression count(*) is given the alias NOBATCHES and expression max(stdate) is given the alias LASTDATE. While referring to view, we have to use these aliases to refer to the values of corresponding expressions.

The following query will display course code and most recent batches starting date using COURSECOUNT view.

```
select  ccode, lastdate from coursecount;

CCODE LASTDATE
----- ---------
asp   15-JAN-01
c     20-JAN-01
java  05-APR-01
ora   15-AUG-01
vbnet 12-JUL-01
xml   02-MAR-01
```

The following query will display the courses for which we have started more than one batch so far.

```
select * from coursecount where  nobatches > 1;

CCODE NOBATCHES LASTDATE
----- --------- ---------
ora           2 15-AUG-01
```

Note: *Though a column is derived from group function, it can be used with WHERE clause if it is part of the view.*

Presenting data in different forms using view

The following example demonstrates how views can be used to present the data of a table in different forms. FACULTY table contains the name of the faculty in NAME column. Now for some application, we have to split the name of the faculty into first name and last name. This task can be achieved using a view. No need to restructure the table.

The following view presents name of the faculty as first name and last name.

```
create view faculty_names
 as
 select  fcode, substr(name,1, instr(name,' ') - 1 ) firstname,
         substr(name, instr(name,' ')+ 1) lastname
 from  faculty;
```

Now it is possible to get the name of the faculty in two parts – firstname and lastname.

```
select * from faculty_names;

FCODE FIRSTNAME                           LASTNAME
```

```
----- ------------------------------ -----------------------------
-
gk    George                         Koch
da    Dan                            Appleman
hs    Herbert                        Schildt
dh    David                          Hunter
sw    Stephen                        Walther
kl    Kevin                          Loney
jj    Jamie                          Jaworski
jc    Jason                          Couchman
```

Now the task of getting the details of faculty where lastname contains more than 5 characters has become easy as shown below.

```
SQL> select * from faculty_names where length(lastname) > 5;

FCODE FIRSTNAME                      LASTNAME
----- ------------------------------ -----------------------------
-
da    Dan                            Appleman
hs    Herbert                        Schildt
dh    David                          Hunter
sw    Stephen                        Walther
jj    Jamie                          Jaworski
jc    Jason                          Couchman
```

Isolating application from changes in definition of table

This is another important application of a view. View can be used to isolate applications from the structure of the tables. This is achieved by creating a view on the required tables and using the view instead of the base table. The advantage with this is that if the structure of the tables is ever changed then we just need to recreate the view to continue to provide the same information as before the change to structure of the base tables.

So that though the structure of the table is ever modified, the application will not be effected by the change as application uses a view to get the required data.

Lets us see a small example to understand what I mean by isolating application from structure of the base tables.

We want to access BCODE, CCODE, STDATE and FEE of each batch. This can be done by using a query to join BATCHES and COURSES tables as follows:

```
select  bcode, b.ccode, fee, stdate
from  batches b, courses c
where  b.ccode = c.ccode;

BCODE CCODE       FEE STDATE
----- ----- --------- ---------
b1    ora        4500 12-JAN-01
b2    asp        5000 15-JAN-01
b3    c          3500 20-JAN-01
b4    xml        4000 02-MAR-01
b5    java       4500 05-APR-01
b6    vbnet      5500 12-JUL-01
b7    ora        4500 15-AUG-01
```

This is fine but what if FEE column is moved from COURSES table to BATCHES table. Then the query needs to access only BATCHES table and COURSES table is not required. So the query is to be rewritten as follows:

```
select  bcode, ccode, fee, stdate
from  batches;
```

That means the application is to be modified to rewrite the query. The column FEE of COURSES table may have been used in several places and now we have to modify all those commands.

How a view can solve the problem? Instead of directly accessing tables, it is possible to create a view on the required tables. For example, we create a view to get BCODE, CCODE, FEE and STDATE from COURSES and BATCHES table as follows:

```
create view  batchdetails
 as
 select  bcode, b.ccode , fee, stdate
 from  batches b, courses c
 where b.ccode = c.ccode;
```

Then we access the data using this view as follows:

```
select  * from batchdetails;
```

If column FEE is removed from COURSES table and placed in BATCHES table, then we recreate the view to access only BATCHES table to get the required data.

```
create or replace view  batchdetails
as
select  bcode, ccode , fee, stdate
from  batches;
```

Then accessing the view BATCHDETAILS will give the same data in spite of a change in the structure of underlying tables. That means the application will have to give as SELECT command to get the information as follows:

```
select  * from batchdetails;
```

However, internally, Oracle is using only BATCHES table to get the data. All this process will be hidden from user and he is aware of only the presence of the view. As you can see, that this operation will make the programs totally independent of the structure of the database. Because even the structure of the tables changes, the view built on the tables are to be rebuild to reflect new structure of the tables. But that doesn’t effect the application.

Storage of view

Oracle stores the query given at the time of creating view in data dictionary. Whenever the view is referred, Oracle executes the query and retrieves the data from the base tables. Whatever data is retrieved that is provided as the data of the view. Since all that Oracle stores regarding view is only the query, a view is also called as **stored query**.

A view may be based on tables or other views or even snapshots (a replica of remote data).

Views and dependence

A view is dependent on base tables. Base tables may be either real tables or again views. Whether a view is valid or not depends on the availability of the base tables. The following examples will illustrate the dependency.

Assume we have a table T1 created as follows:

```
create table t1
(  c1 number(5),
   c2 number(5)
);
```

Let us now create a view on this table as follows:

```
create view v1
as
select * from t1;
```

Oracle stores the query in data dictionary for view. When you use * in query, Oracle expands it to include all columns of the tables. The following query shows the query stored by Oracle in data dictionary.

```
SQL> select  text
  2  from  user_views
  3  where  view_name = 'V1';

TEXT
-------------------------------------------------------------------
---
select "C1","C2" from t1
```

Note: *We must use uppercase letter for object name while querying data dictionary as Oracle stores object name in uppercase.*

Now let us see what happens if you drop the table T1.

```
drop table t1;
```

The view still exists and the definition of the table will remain intact. But the view cannot be used as the base table is not existing. So referring to view will displays the following error message.

```
SQL> select * from v1;
select * from v1
              *
ERROR at line 1:
ORA-04063: view "BOOK.V1" has errors
```

Oracle marks views that are dependent on the base table that is dropped as **Invalid**. You can get the status of any object using USER_OBJECTS table.

```
select status from user_objects where  object_name = 'V1';

STATUS
-------
INVALID
```

However, if you recreate the table and then try to refer to the view the view will be compiled and status of the view is changed to **valid** again.

```
create table t1 ( c1 number(5), c2 number(5), c3 number(5));
```

At the time of recreating the table Oracle just checks whether columns C1 and C2 are present and doesn't mind if base table contains some new columns.

After the table is recreated, if you refer to view then Oracle will try to recompile the view and as it finds out that all the required columns are in the base table it will validate the view. You can get the status of the view using USER_OBJECTS again.

```
select status from user_objects where  object_name = 'V1';

STATUS
-------
VALID
```

In the above if table is recreated but if either column C1 or C2 is present the view cannot be made valid.

Note: *The data types of the columns in the base table do not matter at the time of compiling the view. That means in the above example even columns C1 is of VARCHAR2 type in table T1 still the view will be made valid. Because Oracle precisely looks for columns C1 and C2 in table T1and not for any specific data type.*

Changing Base Table Through View

Though view is generally used to retrieve the data from base table, it can also be used to manipulate base table.

However, not every view can be used to manipulate base table. To be updatable a view must satisfy certain conditions. We will discuss when and how a view can be used to update base table.

The following manipulations can be done on base table through view:

- Delete rows from base table
- Update data of base table
- Insert rows into base table

If a view is to be inherently updatable, then it must not contain any of the following constructs in its query:

- A set operator
- A DISTINCT operator
- An aggregate function
- A GROUP BY, ORDER BY, CONNECT BY, or START WITH clause
- A collection expression in a SELECT list

- A subquery in a SELECT list
- Joins (with some exceptions)

If the view contains columns derived from pseudo columns or expression the UPDATE command must not refer to these columns.

Let us understand how to manipulate the base table through view.

Create a view called ORABATCHES as follows:

```
create view orabatches
as
select bcode, stdate, enddate, timing from batches where ccode =
'ora';
```

As this view doesn't violate any of the conditions mentioned above, it can be used to delete and update base table as follow. The following DELETE command will delete the row from base table – BATCHES – where batch code is b7.

```
delete from orabatches where bcode = 'b7';
```

It is also possible to update the ENDDATE of batch B1 as follows:

```
update orabatches set  enddate = sysdate where bcode ='b7';
```

Updating join views

It is also possible to update a join view (a view that has more than one base table), provided the following conditions are true.

- The DML operations must manipulate only one of the underlying tables.
- For UPDATE all columns updated must be extracted from a key-preserved table.
- For DELETE the join must have one and only one key-preserved table.
- For INSERT all columns into which values are to be inserted must come from a key-preserved table. All NOT NULL columns of the key-preserved table must be included in the view unless we have specified DEFAULT values for NOT NULL columns.

Key-preserved table

A table is key preserved if every key of the table can also be a key of the result of the join. So, a key-preserved table has its keys preserved through a join. To understand what is a key-preserved table, take the following view.

```
create view month_payments
as select p.rollno, name, dp, amount from  payments p, students s
where  p.rollno = s.rollno;
```

In the above view, table PAYMENTS is key-preserved table. So it is possible to make changes to the table even though the view is based on two tables.

The following UPDATE command will update AMOUNT columns.

```
update  month_payments set  amount = 4500
where rollno = 1 and trunc(dp) ='10-jan-01';
```

However, it is not possible to change NAME as follows:

```
update  month_payments set  Name='Koch George'
where rollno = 1;

update  month_payments set  Name='Koch George'
                            *
ERROR at line 1:
ORA-01779: cannot modify a column which maps to a non key-
preserved table
```

Column NAME doesn't belong to key-preserved table. So it is not updatable.

Getting information about updateable columns

It is possible to get the names of the columns that can be modified in a view using USER_UPDATABLE_COLUMNS data dictionary view.

```
select column_name, updatable, insertable, deletable from
user_updatable_columns
where  table_name = 'MONTH_PAYMENTS'

COLUMN_NAME                    UPD INS DEL
------------------------------ --- --- ---
ROLLNO                         YES YES YES
NAME                           NO  NO  NO
DP                             YES YES YES
AMOUNT                         YES YES YES
```

WITH CHECK OPTION

This option is used to prevent updations and insertions into base table through view that the view cannot later retrieve.

Let us create a view that retrieves information about payments where amount paid is more than 3000.

```
create view high_payments
as
select * from payments
where  amount > 3000;
```

The following is the data that is retrieved by the view.

```
select * from high_payments;

   ROLLNO DP           AMOUNT
--------- --------- ---------
        1 10-JAN-01      4500
        2 11-JAN-01      3500
        5 16-JAN-01      5000
        6 14-JAN-01      3500
        7 15-JAN-01      3500
       10 10-APR-01      4500
       11 10-APR-01      3500
```

As the view is updatable, Oracle allows any change to be made to the view. For example the following UPDATE statement will change the amount paid by student 11 to 2500.

```
update high_payments
set amount = 2500
where  rollno = 11;
```

After the change is made if we try to retrieve the data from the view, it will NOT retrieve row that we have updated as it no longer satisfies the condition – AMOUNT > 3000.

```
select * from high_payments;

   ROLLNO DP           AMOUNT
--------- --------- ---------
        1 10-JAN-01      4500
        2 11-JAN-01      3500
        5 16-JAN-01      5000
        6 14-JAN-01      3500
        7 15-JAN-01      3500
       10 10-APR-01      4500
```

That means view allows the changes even though the changes will make the rows of the base table irretrievable after the change. However, Oracle allows you to prevent such changes by using WITH CHECK OPTION option at the time of creating view.

The following is the modified CREATE VIEW command to create HIGH_PAYMENT view. It uses WITH CHECK OPTION to make sure that all updations and insertion that are made to the base table are retrievable by the view.

```
create or replace view high_payments
as
select * from payments
where  amount > 3000
with check option
constraint high_payment_wco;
```

The above command replaces the existing view with the new version. That is the reason why we used OR REPLACE option of CREATE VIEW command.

The command added WITH CHECK OPTION and assigned a name to that constraint. If we do not give any name then Oracle automatically assigns a name to this constraint.

Now let us see what happens if we try to insert a row into the base table – PAYMENTS through HIGH_PAYMENTS view with 2000 as the value for AMOUNT column.

```
SQL> insert into high_payments values (11,sysdate, 2000);
insert into high_payments values (11,sysdate, 2000)
            *
ERROR at line 1:
ORA-01402: view WITH CHECK OPTION where-clause violation
```

As you can see, WITH CHECK OPTION doesn't allow the insertion since the condition given in the query of the view is not satisfied by the data given in insertion.

However, it is possible to insert the same row directly to base table – PAYMENTS, but it cannot be inserted through the view HIGH_PAYMENTS if view cannot retrieve row after the insertion.

Dropping a View

DROP VIEW command is used to drop a view. Dropping a view has no effect on the base tables on which view is based. Users who were accessing the data of the base tables using view will no longer be able to access the data of the base tables.

```
DROP VIEW view_name;
```

Summary

A view is an object in Oracle database, used to provide access only to the required portion of the table by hiding the remaining part of the table. A view can simplify a query by storing complex query in the database. A view doesn't contain any data and retrieves the data from the base table whenever it is referred.

As views are used just like tables they are called as **virtual tables.** It is also possible to manipulate base table through view. Manipulations like INSERT , DELETE and UPDATE can be performed by view provided it fulfills the conditions set by Oracle.

Exercises

1. What are the major applications of a view?
2. A view can be used with ALTER TABLE command [T/F] ?________ .
3. The table on which a view is based is called as ______.
4. When a table is dropped then all the views based on it will be dropped automatically [T/F]? _______.
5. A view can be used to manipulate base table [T/F]? ______.
6. Create a view, which contains the course code and number of students who have taken that course so far.
7. Create a view to contain the following: bcode, course name, faculty name, stdate, enddate and no. of days between enddate and stdate for all completed batches.
8. Create a view to get bcode, ccode, fcode, timing , stdate and enddate for all completed batches. Also ensure the changes made to base table through view are retrievable through view.