# 8. Design Views

## 8.1 Create views to store queries in the database

<http://docs.oracle.com/cd/B19306_01/server.102/b14200/statements_8004.htm>

### What is a view?

* A view is a data object that contains no data of its own – a virtual table whose contents are taken from other tables ( or other views) through the execution of a query
* The tables we have created so far contain data persistenly stored in the database ( a permanent base table/ source table, saved, etc)
* Once created, views can be used like a table – for SELECT, INSERT, etc. – The query that makes up the view is executed whenever the view is used in such a statement
* Can selectively hide part of the data from users

### Advantages of Views

* Security
  + Hide things (columns ) from users
  + Allow the system to give different permissions to the same table
* Simplicity
  + Display results of computed columns
  + Simplify queries and hide SQL syntax
* Extendibility
  + Allow the system to change without breaking how the application works – underlying table changes, but as far as the user is aware, the view stays the same.
* Maintain Data Integrity
  + Allow the system to have different processing rules (via triggers) for the same table.

### Disadvantages of Views

* **Performance – Do not notice if it is bad performance, might forget that the view may be doing a lot of work.**
  + Complex queries can be run on views. Views can be created with complex queries. Combining the two can result in poor performance.
  + Since the complexity of the view is hidden, users are not aware of the additional complexity they are creating by running queries.
* **Manageability**
  + Views make it easy to create many different layers adding to the complexity of the database. This makes debugging problems difficult.
* **Update Restrictions**
  + More complex views and nested views will possibly create read-only views.

### How Views work.

* Essentially a view is just a query with a name.
* When you call or use a view, you are accessing the data stored in the underlying tables.
* When a view is created, the SELECT statement is not actually run at that time – the SELECT statement gets stored and is run when called usually within another SELECT statement.

### CREATE VIEW Syntax

* The CREATE VIEW statement:

CREATE [OR REPLACE] VIEW viewname [ ( aliases for columns ) ] AS

SELECT statement [WITH { READ ONLY | CHECK OPTION } ]

* Use OR REPLACE if you want to replace an existing view with a new select statement
* Provide aliases for columns if you don't want the column names from the select statement
* WITH READ ONLY means only selects are allowed against the view; updates, inserts and deletes are not allowed.
* WITH CHECK OPTION means the user can insert, update, and delete, but only those rows that are visible in the VIEW; without the check option, the user could create a row that would not appear in the view.
* Naming columns in views – can give names to use for the columns in a list after the view name. Alternatively, can use column aliases in the SELECT statement.
* A view's SELECT statement can be pretty much any legal SELECT, but generally do not include an ORDER BY clause. Some systems may also restrict the ability to create views with UNION, etc.
  + Views can have joins, subqueries
  + Views can be derived from other views
* Can use SELECT \* as a view, if the underlying table changes, the view will likely change how it works, which is not good.
* Beware the effect of using SELECT \* … in a view – if the table definition changes, the view many behave unexpectedly

### DROP VIEW Syntax

* To get rid of a view: DROP VIEW:

DROP VIEW viewname;

### Changing a View's definition

* To change the definition of a view, use CREATE **OR REPLACE** VIEW.
* ALTER VIEW (in Oracle) is only for "re-compiling" the view.

Once a view is created, it can be used like a table. Query the view as you would a table. With restrictions, you may be able to modify the data underlying the view.

## 8.2 Describe circumstances under which data can be updated through a view

### Modifying data through views

* Commands to modify data through a view may not be understood by SQL in an unambiguous way, so there are some occasions when data cannot be changed through a view.

Base table(s): Original tables that the view is created from.

Key preserved table: A row from the base table will appear AT MOST once in the result output from the view (The data won’t be repeated). It also means that every key from the base table can be a key of the resulting join. (the keys of the key preserved table are also keys for the view). It is called a Key-preserved table because all its keys are preserved through the join. Look at the result set and compare it to your base tables to determine the key preserved table. You don’t always need all the key columns of a table to be selected in the view for the table to be Key persevered but you do need to ensure that the values show up only once in the result set.

The key preserved table will generally be on the “Many” side of a one-to-many relationship (often is a joining table) – unless it is by itself in the view



Code:

SQL> CREATE VIEW emp\_dept AS

SELECT a.empNo, a.eName, a.salary, a.deptNo, b.dName

FROM Employee a, Department b

WHERE a.deptNo = b.deptNo;

View created.

SQL> SELECT \* FROM emp\_dept;

EMPNO ENAME SALARY DEPTNO DNAME

--------- ---------- --------- --------- --------------

7369 SMITH 5000 20 RESEARCH

7499 ALLEN 2129.6 30 SALES

7521 WARD 1663.75 30 SALES

7566 JONES 3959.73 20 RESEARCH

7654 MARTIN 1663.75 30 SALES

7698 BLAKE 3793.35 30 SALES

7782 CLARK 3260.95 10 ACCOUNTING

7788 SCOTT 3993 20 RESEARCH

7839 KING 6655 10 ACCOUNTING

7844 TURNER 1996.5 30 SALES

7876 ADAMS 1464.1 20 RESEARCH

7900 JAMES 1264.45 30 SALES

7902 FORD 3993 20 RESEARCH

7934 MILLER 1730.3 10 ACCOUNTING

14 rows selected.

In the above example Employee is key preserved table. Rows from EMPLOYEE appears only once. DEPTARTMENT is not a key preserved table. DEPTNO is key column in dept. But it is not a key column in EMP\_DEPT view (DEPTNO repeats in the results).

ANSI rules – A view is not updatable if it has any of the following items in the select clause of the View:

* DISTINCT
* Any aggregate functions or analytical (SUM,MIN,AVG,etc)
  + Note: Computed expression(string literal, salary \*1.1, etc) cannot be changed, but the other columns can be changed
* GROUP BY, ORDER BY, HAVING clause
* References to more than one table (FROM clause, subquery, UNION, etc)
  + Oracle supports updateable join views, with restrictions
    - Can only update one base table at a time
    - Must have only one key preserved table
* If a view references another view, that is not updatable, the new view is not updatable

Rules per Oracle Database 11g: The Complete Reference  
<http://library.books24x7.com/book/id_26827/viewer.asp?bookid=26827&chunkid=498234295>

If a view is based on a single underlying table, you can **insert**, **update**, or **delete** rows in the view. This will actually **insert**, **update**, or **delete** rows in the underlying table. There are restrictions on your ability to do this:

* You cannot **insert** if the underlying table has any **NOT NULL** columns that don't appear in the view.
* You cannot **insert** or **update** if any one of the view's columns referenced in the **insert** or **update** contains functions or calculations. Note: Can make changes to non calculation columns.
* You cannot **insert**, **update**, or **delete** if the view contains **group by**, **distinct**, or a reference to the pseudo-column RowNum.

You can **insert** into a view based on multiple tables if Oracle can determine the proper rows to insert. In a multitable view, Oracle determines which of the tables are *key-preserved*. If a view contains enough columns from a table to identify the primary key for that table, the key is preserved and Oracle may be able to insert rows into the table via the view.

**Bottom line: For a view to be updateable, the DBMS must be able to trace any row or column back to its row or column in the source table.**

If a view isn't updatable, then it is read only. Views that are not updatable can be modified using INSTEAD OF triggers.

### Creating copies of data

* Rather than create a view, which doesn't make a copy of the data, you may want an independent copy of some data stored in a separate table.

#### Useful for

* Creating test tables with sample data
* Extracting info for static reporting (a snapshot), when the querying process is time consuming.

#### Two options for copying data/tables:

* + Create the table, then use the INSERT … SELECT … syntax

INSERT INTO newTable  
 SELECT \* FROM oldTable;

INSERT INTO newTable (columns)

SELECT columns

FROM oldTable;

* + Use the CREATE TABLE … AS SELECT … syntax

CREATE TABLE newTABLE AS

SELECT Columns  
 FROM oldTable;

### Materialized Views

A materialized view differs from an ordinary view because it caches the query results in a concrete table. This table may be updated from the original base tables from time to time. Often referred to as a snapshot. –NOTE: some db’s do not support this (access)

#### Advantages:

* More efficient access, for users at least

#### Disadvantages:

* Some of this data may be out of date
* Updates done to the materialized view are usually logged and performed on the underlying tables on either a scheduled basis or when the view is dropped

#### Used when:

* In a data warehousing scenario, where frequent queries of the actual base tables can be very slow.
* The queries that define the view are complicated / slow