**Views**

The kinds of tables that you have been dealing with up until now are called base tables. These are tables that contain data. There is another kind of table, however: the view. Views are tables whose contents are taken or derived from other tables. They are operated on in queries and DML statements just as base tables are, but they contain no data of their own.

**CREATE VIEW Londonstaff AS SELECT \* FROM Salespeople WHERE city = 'London';**

You now own a view called Londonstaff. You can use this view just like any other table. It can be queried, updated, inserted into, deleted from, and joined with other tables and views.

**SELECT \* FROM Londonstaff;**

When you told SQL to SELECT all rows from the view, it executed the query contained in the definition of Londonstaff, and returned all of its output. Had there been a predicate in the query of the view, only hose rows of the view that satisfied it would have been output.

Views greatly extend the control you have over your data. They are an excellent way to give people access to some but not all of the information in a table. If you wanted your salespeople to be able to look at the Salespeople table, but not to see each other's commissions, you could create a view for their use (with the following statement:

**CREATE VIEW Salesown AS SELECT snum, sname, city FROM Salespeople;**

UPDATING VIEWS

This view can now be modified by DML update commands, but the modifications will not affect the view itself. They will be passed along to the underlying table:

**UPDATE Salesown SET city = 'Palo Alto' WHERE snum = 1004;**

The effect of this is identical to performing the same command on the Salespeople table. However, if a salesperson tried to UPDATE his commission

**UPDATE Salesown SETcomm = .20 WHERE snum = 1004;**

it would be rejected, because there is no comm field in the Salesown view. It is important to note that not all views can be updated.

The syntax to eliminate a view from the database is similar to that for removing base tables:

**DROP VIEW <view name>**

UPDATING VIEWS

One of the most difficult and ambiguous aspects of views is the implication of their usage with the DML update commands. these commands actually affect the values in the underlying base tables of the view. This is something of a contradiction. A view consists of the results of a query, and when you update a view, you are updating a set of query results. But the update is not to affect the query per set; it is to affect the values in the table(s) on which the query was made, and thereby change the output of the query.

The criteria that determine whether or not a view is updatable in SQL are as follows:

• It must be drawn on one and only one underlying table.

• It must include the primary key of that table (this is not technically enforced by the ANSI standard, but you would be well-advised to stick to it).

• It must have no fields that are aggregate functions.

• It must not specify DISTINCT in its definition.

• It must not use GROUP BY or HAYING in its definition.

• It must not use subqueries (this is an ANSI restriction that is not enforced in some implementations).

• It may be defined on another view, but that view must also be updatable.

Examples:

**View creation from Single table:**

Consider the CUSTOMERS table having the following attributes:

ID, NAME, AGE, ADDRESS, SALARY

**To create a view from CUSTOMERS table with customer name and age:**

CREATE VIEW CUSTOMERS\_VIEW AS

SELECT name, age

FROM CUSTOMERS;

**Now, you can query CUSTOMERS\_VIEW in similar way as you query an actual table as follows.**

SELECT \* FROM CUSTOMERS\_VIEW ;

**With CHECK Option:**

CREATE VIEW CUSTOMERS\_VIEW AS

SELECT name, age

FROM CUSTOMERS

WHERE age IS NOT NULL

WITH CHECK OPTION;

**Consider the following two tables:**

The attributes of EMP table are EMPNO, ENAME, JOB, MGR, HIREDATE, SAL, COMM, DEPTNO.

The attributes of DEPT table are DEPTNO, DNAME, LOC.

**The following statement creates the emp\_dept view:**

CREATE VIEW emp\_dept AS SELECT emp.empno, emp.ename, emp.deptno, emp.sal, dept.dname, dept.loc FROM emp, dept WHERE emp.deptno = dept.deptno AND dept.loc IN ('DALLAS', 'NEW YORK', 'BOSTON');

**UPDATE Statements**

UPDATE emp\_dept SET sal = sal \* 1.10 WHERE deptno = 10;

**DELETE Statements**

DELETE FROM emp\_dept WHERE ename = 'SMITH';

**INSERT Statements**

INSERT INTO emp\_dept (ename, empno, deptno) VALUES ('KURODA', 9010, 40);

**DROPPING Views**

DROP VIEW emp\_dept;