SQL Course

Introduction to Views

- To introduce the concept of a view defined on one or more tables.
- To point out the possible advantages and disadvantages of using views.
- To demonstrate how the RDBMS processes views.

Sometimes in practise the user of a database may want to reference data regularly from different tables and see this data on screen at the same time. This can be achieved using a select statement by selecting the data the user wants from the tables in question. If the same data is being accessed regularly then one could ask the question: " Why not create a table with the data that the user wants as the columns in the table?".

- The problem with this is that each time data is added or updated to the original tables the new table must be checked to see if this data should be added or updated in the new table. This causes a degradation in performance.
- Another potential problem is that if another user wants to see the data in a different format, another table must be created and maintained, adding redundancy and making the problem even worse.

To provide the desired functionality without the drawbacks that were just mentioned, SQL provides a mechanism that enables you to create virtual tables, called views, that can be queried as regular tables but not stored as such. The contents of these virtual tables change when the contents of the regular tables on which they are defined change. This happens logically: updates are made on regular, stored tables and the system computes dynamically the contents of a view, when needed. Since the view mechanism recomputes the result when it is accessed , it may be slower than using a stored table which materialises the view and which is kept up to date at each update.

When you become familiar with the **SELECT** statement in later modules you will see the real power and benefits of using views in the relational database model. Creating a view based on a **SELECT** statement gives you an easy way to examine and handle just the data you need - no more, no less. A view can be used in an SQL statement just as if it were a table. This means that a view can be used in other SQL statements and can even be used to create other views. The rows and columns of data visible through the view are the query results produced by the query that defines the view.

Advantages:

- It saves the user from having to type in long queries each time they wish to see a subset of data across a number of tables.
- Views can be used in other SQL statements as if it were a table in the original database.
- Views allow different users to focus in on particular data and tasks of interest or concern. No extraneous or distracting information gets in the way. Using views, the database can be customised and so tailored to suit a variety of users with dissimilar interests and skill levels.
- Views provide security by hiding sensitive or irrelevant parts of the database. Access privileges can be granted and revoked on views as they would be on normal base tables.
- A view can be seen to be independent from underlying changes in structure to the original

tables. This means that if base tables are changed the view can be re-composed by altering the **SELECT** statement that was used to create the view.

- Views simplify database access by presenting the structure of the stored data in the way that is most natural for each user.
- A view can simplify multi table queries by drawing data from several tables and presenting them in a single table, thus turning the multi table query into a single table query against the view.
- If data is accessed and entered through a view, the DBMS can automatically check the data to ensure that it meets specified integrity constraints.

Disadvantages:

- As the database must translate queries against the view to queries against the source tables it means a degradation of performance and if a view is defined by a complicated query it may take a long time to complete.
- When a user tries to update the rows of a view, the database must translate the request into an update on rows of the underlying source tables. This is possible for simple views, but more complex views cannot be updated.

How it Works

When the database encounters a reference to a view in a SQL statement, it finds the definition of the view stored in the database. The database then translates the request that references the view into an equivalent request against the source tables of the view and carries out the equivalent request. In this way the database maintains the illusion of the view while maintaining the integrity of the source tables. For simple views the database will draw the rows of the view on the fly but for more complex views the database will first need to store the results of the query that defines the view in a temporary table. The database fills your requests for view access from this temporary table and discards the table when it is no longer needed.