

In addition to the propagation of data modifications, design changes also propagate. If a user changes the design of a View, such as specifying a new selection criterion, that View is updated as well as all views dependent upon that View. Again, this is all done automatically, so that the user is unaware of the process.

Contrast this approach to that of a dynaset or "live" answer table. Although a dynaset includes storage having pointers to corresponding records in the base table, the dynaset is not updated automatically when a change is made to the base table. Moreover, the dynaset approach is ill-suited for use as a live view. In particular, that approach would require re-executing the query for each modification to the base table. The resource and time requirement for re-executing the query makes this approach impractical. The incremental maintenance approach of the present invention, in contrast, handily solves the problem of providing a live view, yet does not entail resource-impractical operations (e.g., re-executing queries for each modification).

FIG. 19 is a flowchart 1900 illustrating the method of the present invention for incremental maintenance of views. At step 1901, a user modification occurs, such as the user changing data in a field of a particular record. At step 1902, the method will follow a particular branch of logic, based on whether the user modification was an "Insert," "Delete," or "Modify" operation. The remaining steps of the method will be illustrated by following the "Modify" case arm, which includes elements of both "Insert" and "Delete."

The document which was modified has a list of views (Doc List) which are dependent on it. These dependents are notified at step 1903. Step 1904 illustrates notification of a particular view. Here, the View is notified what operation type occurred (e.g., Insert, Delete, or Modify), as well as identification (e.g., key and/or record number) of the affected record.

At step 1905, the method checks whether the modified record still satisfies the View constraints. If "Yes" at this step, then at step 1906 the method loops back to step 1904 for processing of other remaining views; if no other views remain at this point, the method is done. In other words at step 1905, if the record still satisfies the View constraint then that particular View need not be changed (other than refreshing the particular data value which has changed). If, on the other hand, the record does not still satisfy the View constraint at step 1905, then the method proceeds to step 1907 where the View (which itself is stored as a tree) deletes the pointer to the record, thereby removing it from the View. Thus, step 1907 leads to deletion of the entry in the View (i.e., B-tree pointer) which corresponds to the record. At the completion of this step, the method will loop back to 1904 for any other remaining views, or terminate in the case of no remaining views.

Consider a user modification which keeps a particular record in a View yet changes the associated record geometry, such as an operation where the user adds a substantial amount of text to a text field of a particular record. Recall that the View stores in its B-tree the geometry associated with the record. If this geometry is modified (e.g., record grows larger), the B-tree is updated with the new geometry. More particularly, the path which leads to this particular record is recalculated using the new geometry for the record. In a corresponding manner, the stored geometry at each level in the B-tree is updated to reflect the change. Since only the path for the record is refreshed, the process is particularly efficient—there is no need to globally recalculate the effect on the View.

In a like manner, if a record is inserted or deleted, the geometry along the path associated with the record is refreshed accordingly. Again, the operation is particularly efficient, since only a single path through the tree need be traversed.

While the invention is described in some detail with specific reference to a single preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. Thus, the true scope of the present invention is not limited to any one of the foregoing exemplary embodiments but is instead defined by the appended claims.

What is claimed is:

1. In a computer system having a database storing a plurality of data records, a method for creating a "live" report of said database, the method comprising:

(a) for each data record, storing with the database information describing record geometry required for presenting each said data record in the report;

(b) receiving a request for displaying a certain page of the report;

(c) converting said request for displaying a certain page of the report into a request for displaying a portion of the report located at a certain distance from one end of the report wherein said certain distance is computed from: (page number for said certain page) × (distance of printable area of each page);

(d) determining from said stored record geometries which particular data records fall on said certain page; and

(e) presenting the report to a user by rendering on a display device said particular data records determined in step (c) to fall on said certain page.

2. The method of claim 1, wherein said record geometry comprises volume required for displaying a particular record in the report.

3. The method of claim 1, wherein said record geometry comprises a height required for displaying a particular record in the report.

4. The method of claim 1, wherein step (a) includes:

storing a B-tree with said database, said B-tree having leaf nodes pointing to particular database records in the database, and wherein each node in the B-tree stores an accumulated record geometry for records of nodes beneath it.

5. The method of claim 4, wherein each node in the B-tree also stores an accumulated record count for records of nodes beneath it.

6. The method of claim 4, wherein step (c) includes:

determining which record lies at said certain page by:

(i) traversing said B-tree for accumulating record volumes stored at nodes of the B-tree, and

(ii) upon accumulating sufficient record volumes which at least equal a distance to said certain page, reading from the B-tree the record number pointed to by the B-tree at that accumulated volume.

7. The method of claim 6, wherein substep (i) includes: starting from a root node of the B-tree, traversing nodes of the B-tree until accumulated record volumes stored by nodes which have been traversed meet or exceed a volume required to reach said certain page of the report.

8. The method of claim 1, wherein no data record is allowed to straddle a page boundary of the report.

9. The method of claim 1, further comprising:

receiving user input for modifying a particular data record; and

in response to said user input, updating the information describing record geometry required for presenting said particular data record.

10. The method of claim 9, further comprising:

re-determining from the updated stored record geometries which particular data records fall on said certain page; and