DATABASE MANAGEMENT SYSTEMS

Lecture 9 Additional Notes

Views

A view is the dynamic result of one or more relational operations (queries) operating on base relations (tables) to produce another relation (virtual table).

It is a virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request. The contents of a view are defined as a query on one or more base relations.

Views are dynamic, meaning that changes made to base relations (tables) that affect view attributes are immediately reflected in the view.

Purpose of Views

The view mechanism is desirable for the following reasons:

- A view provides a powerful and flexible security mechanism by hiding parts of database from certain users. Users are not aware of any hidden attributes (columns) or rows that are missing from the view.
- A view can be defined over one or more base table(s) with a user being granted the appropriate privileges to use the view, but not to use the base table(s). In this way, a view is more restrictive than simply granting privileges to a user on the base table(s). For example, it provides additional mechanism to restrict access to specific rows and/or columns of the base table(s).
- A view permits users to access data in a customized way, so that same data can be seen by different users in different ways, at same time.
- It can simplify complex operations on base relations. For example, queries involving joins are defined once in a view and the user will then be able to construct simpler queries using the view.
- Views also provide logical data independence. For example:
    - If a new attribute (column) is added to a relation (table), existing users can be unaware of its existence if their views are defined to exclude it (they do not need to see the new column). Logical data independence means there is no need to update the view because a new column has been added to the conceptual schema.
    - If an existing table is rearranged or split up into smaller tables, the view can be updated to include the necessary columns from the smaller tables using a join, so that the users can continue to see their original views. Logical data independence here means that although the conceptual schema has changed, the queries for the external views can be updated such that the user is unaware of any underlying changes.

Advantages and Disadvantages of Views[1]

Advantages

In the case of a DBMS running on a standalone PC, views are usually a convenience, defined to simplify database requests. However, in a multi-user DBMS, views play a central role in defining the structure of the database and enforcing security and integrity. The major advantages of views are described next.

Data independence

A view can present a consistent, unchanging picture of the structure of the database, even if the underlying source tables are changed (for example, if columns added or removed, relationships changed, tables split, restructured or renamed). If columns are added or removed from a table, and these columns are not required by the view, then the definition of the view need not change. If an existing table is rearranged or split up, a view may be defined so that users can continue to see the old table. In the case of splitting a table, the old table can be recreated by defining a view from the join of the new tables, provided that the split is done in such a way that the original table can be reconstructed. We can ensure that this is possible by placing the primary key in both of the new tables.

Currency

Changes to any base tables in the defining query are immediately reflected in the view.

Improved security

Each user can be given the privilege to access the database only through a small set of views that contain the data appropriate for that user, thus restricting and controlling each user's access to the database.

Reduced complexity

A view can simplify queries, by drawing data from several tables into a single table, thereby transforming multi-table queries into single-table queries.

Convenience

Views can provide greater convenience to users as users are presented with only the part of the database that they need to see. This also reduces the complexity from the user's point of view.

Customisation

Views provide a method to customise the appearance of the database, so that the same underlying base tables can be seen by different users in different ways.

Data integrity

If the WITH CHECK OPTION clause of the CREATE VIEW statement is used, then SQL ensures that no rows that fail to satisfy the WHERE clause of the defining query is ever added to any of the underlying base table(s) through the view, thereby ensuring the integrity of the view.

[1] Source: [© Pearson - Database Systems: A Practical Approach to Design, Implementation, and Management - http://app.kortext.com/read/80118/?page=258-259]

Disadvantages

Although views provide many significant benefits, there are also some disadvantages with SQL views.

Update restrictions

In some cases, a view cannot be updated. See today's lecture notes – slides 15 – 17.

Structure restrictions

The structure of a view is determined at the time of its creation. If the defining query was of the form SELECT * FROM …., then the * refers to all the columns from the base table present when the view is created. If columns are subsequently added to the base table, then these columns will not appear in the view unless the view is dropped and recreated.

One of the questions in today's lab tutorial exercise demonstrates this.

Performance

There is a performance penalty to be paid when using a view. In some cases, this will be negligible, while in other cases it may be more problematic. For example, a view defined by a complex, multi-table query may take a long time to process, as the view resolution must join tables together *every time the view is accessed*. View resolution requires additional computer resources. An alternative approach to maintaining views to overcome this is to use view materialisation (or simply temporary tables).

View resolution is the process of looking up the definition of the view and translating the query into an equivalent query request against the source tables before performing this equivalent query request. In view materialisation, the view is stored as a temporary tables in the database when the view is first queried. Thereafter, queries run against the materialised view (temporary table) can be much faster than recompiling the view each time. The difficulty of this alternative approach, however, is in maintaining the currency of the data in the temporary table while the base table(s) are being updated.

Review Questions

- Discuss the advantages and disadvantage of views.
- How can views be used as a security mechanism in a Database?.